AFIT/GAE/ENY/92D-17

1

HELICOPTER FLIGHT CONTROL SYSTEM DESIGN USING THE LINEAR QUADRATIC REGULATOR FOR ROBUST EIGENSTRUCTURE ASSIGNMENT

THESIS

Dempsey D. Solomon, CPT, USA

AFIT/GAE/ENY/92D-17

AFIT/GAE/ENY/92D-17

HELICOPTER FLIGHT CONTROL SYSTEM DESIGN USING THE LINEAR QUADRATIC REGULATOR FOR ROBUST EIGENSTRUCTURE ASSIGNMENT

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

Dempsey D. Solomon, B.S.
Captain, USA

December 1992

Acknowledgements

My thanks to the staff and faculty at AFIT, especially Dr. Bradley S. Liebst, who provided an outstanding environment in which to learn as well as their knowledge and guidance. Also, thanks to Mr. Doug Burkholder who allowed me to learn as little about computers as possible.

I would especially like to thank Cindy, Matthew and A.J. who teach me a little more every day.

Dempsey D. Solomon

TABLE OF CONTENTS

List of Figures

List of Tables

List of Symbols

α	minimum singular value of return difference matrix
δ_a	lateral cyclic control movement
δ_b	longitudinal cyclic control movement
δ_c	collective control movement
δ_p	tail rotor control movement
θ	pitch angle
θ_i	i^{th} eigenvector difference minimization parameter
λ	eigenvalue
λ_a	achievable eigenvalue
λ_d	desired eigenvalue
μ	redefined control vector
ρ	arbitrary constant
ϕ	roll angle
σ	minimum degree of stability
$\underline{\sigma}$	minimum singular value
ω	frequency
∞	infinity
$^\circ$	degrees
*	multiplication
$\int$	integral
Σ	summation
A	state matrix
A_{ND}	non-dimensional state matrix
ACLS	closed loop state matrix
ANEW	redefined state matrix

B	control matrix
B_{ND}	non-dimensional control matrix
det	determinant of a matrix
dt	differential element of time
Ed	diagonal matrix of desired eigenvalues
e	base of natural logarithms
Fe	eigenvalue weighting matrix
Fv	eigenvector weighting matrix
G	redefined regulator gain matrix
I	identity matrix
IGM	independent gain margin
IPM	independent phase margin
i or j	square root of negative one
J	LQR performance index
$\bar{J}$	algorithm performance index
J_λ	eigenvalue contribution to $\bar{J}$
K	feedback gain matrix
kmax	maximum optimization iterations
MHN	matrix used to create weighting matrix QRS
n	number of states
P	Riccati equation solution matrix
p	roll rate
q	pitch rate
Q	LQR state weighting matrix
$\bar{Q}$ or QRS	positive definite weighting matrix formed by Q, R and S

QH	matrix used to create state weighting matrix Q
QNEW	redefined state weighting matrix
r	yaw rate
R	LQR control weighting matrix
RM	matrix used to create control weighting matrix R
r-code	code specifying type of R matrix to be used
S	LQR cross coupling weighting matrix
s-code	code specifying type of S matrix to be used
SN	matrix used to create weighting matrices
tol	convergence tolerance for algorithm
u	control vector
u	longitudinal velocity
v	lateral velocity
v_a	achieved eigenvector
v_d	desired eigenvector
V_d	matrix of desired eigenvectors
w	vertical velocity
x	state vector
$\dot{x}$	derivative of state vector
XGUESS	vector used to minimize performance index
z_i	i^{th} element of vector z
Z_{ij}	element in i^{th} row and j^{th} column of matrix Z
Z^T	transpose of matrix Z
Z^{-1}	inverse of matrix Z
Z^*	Hermitian transpose of matrix Z

AFIT/GAE/ENY/92D-17

Abstract

This thesis applies modern, multi-variable control design techniques, via a FORTRAN computer algorithm, to U.S. Army helicopter models in hovering flight conditions. Eigenstructure assignment and Linear Quadratic Regulator (LQR) theory are utilized in an attempt to achieve enhanced closed loop performance and stability characteristics with full state feedback. The addition of cross coupling weights to the standard LQR performance index is specifically addressed. A desired eigenstructure is chosen with a goal of reduced pilot workload via performance characteristics and modal decoupling consistent with current helicopter handling qualities requirements. Cross coupling weighting is shown to provide greater flexibility in achieving a desired closed loop eigenstructure. Also, while the addition of cross coupling weighting is shown to eliminate stability margin guarantees associated with LQR methods, the study shows that the modified algorithm can achieve a closer match to a desired eigenstructure than previous versions of the program while maintaining acceptable stability characteristics.

HELICOPTER FLIGHT CONTROL SYSTEM DESIGN USING THE LINEAR QUADRATIC REGULATOR FOR ROBUST EIGENSTRUCTURE ASSIGNMENT

I. Introduction

This thesis applies modern, multi-variable control design techniques, via a FORTRAN computer algorithm, to U.S. Army helicopter models in hovering flight conditions. Eigenstructure assignment and Linear Quadratic Regulator (LQR) theory are utilized in an attempt to achieve enhanced closed loop performance and stability characteristics with full state feedback. The addition of cross coupling weights to the standard LQR performance index is specifically addressed. A desired eigenstructure is chosen with a goal of reduced pilot workload via performance characteristics and modal decoupling consistent with current helicopter handling qualities requirements. This work builds upon previous efforts by Robinson [1] and Huckabone [2] at the Air Force Institute of Technology (AFIT), the major difference being the addition of cross-coupling weighting in the LQR performance index.

Helicopters

Helicopters serve as a good platform for the application of automatic control. The aerodynamic and structural design of rotary winged aircraft create inherent problems with regard to stability and control. In order to apply the tools of control design, a mathematical model of the system is needed. McRuer [3:chap 4] presents the assumptions and techniques used to represent an aircraft as a set of linear differential equations with constant coefficients, or equations of motion. Reid [4:chap 6] provides the techniques by which these equations can be represented as transfer functions or via the state space model

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{1}$$

The state space model is used for the design technique presented in this thesis.

Two helicopter models are used in this thesis. One is based on the AH-64 (Apache) attack helicopter [5] the other is the UH-60 (Blackhawk) utility helicopter [6]. Both aircraft are considered to be conventional helicopters in that they have a single main rotor to produce primary lift and a single tail rotor to produce anti-torque and directional control forces. The models used here represent the aircraft in hovering flight.

The state vector x in equation (1) is the column vector comprised of the following respective states:

u - longitudinal (long) velocity (vel) [feet/second]

v - lateral (lat) velocity [feet/second]

w - vertical velocity [feet/second]

p - roll rate [radians/second]

q - pitch rate [radians/second]

r - yaw rate [radians/second]

ϕ - roll angle [radians]

θ - pitch angle [radians]

The input vector **u** is the column vector comprised of the following respective inputs:

δ_c - collective

δ_b - longitudinal cyclic

δ_a - lateral cyclic

δ_r - tail rotor

Collective input refers to a simultaneous change in the angle of attack of all the main rotor blades that causes a change in magnitude of the lift produced by the main rotor. Cyclic input refers to independent angle of attack changes of the blades that result in a change of main rotor lift direction. Tail rotor input is similar to collective input applied to the tail rotor. For the AH-64 model, the inputs are expressed in degrees of blade movement, while the UH-60 inputs are expressed as inches of control movement. Standard sign conventions for states and controls are used as presented in reference [3].

The **A** and **B** matrices of equation (1) contain the constants of the equations of motion. These matrices are presented in appendices A and B for the AH-64 and UH-60, respectively. The AH-64 model has been non-dimensionalized as described by Huckabone [2:52-54].

The purpose of designing an automatic control system for an aircraft is to enhance its handling or flying qualities.

> Flying qualities determine the ease and accuracy with which a pilot can accomplish the various tasks or maneuvers that constitute the aircraft's mission. The elements which directly affect flying qualities are the stability and control characteristics of the aircraft which link the controllers that the pilot manipulates to the aircraft response states that the pilot desires to control [7:2-1].

Factors that can cause adverse flying qualities are: instabilities, sluggish response and uncommanded responses. Specific flying quality requirements for helicopters are set forth in Aeronautical Design Standard-33 (ADS-33) [8]. Analysis of the open loop eigenstructure of the helicopter model will reveal the presence of instabilities and state to state coupling. The B matrix of the helicopter model shows the control-state coupling that is specifically addressed in this thesis.

The need to improve handling qualities arises from the desire to reduce the pilot's workload as much as possible. This is done to allow the pilot to perform tasks, other than flying, in conjunction with the aircraft's mission. Examples of tasks include navigation, communication and weapon delivery. One way that flying qualities may be improved is by the reduction of control-state coupling or cross coupling. Ideally, a helicopter pilot should have the ability to command precise control of the hovering aircraft's position via direct, single axis control. The desired control-state coupling is:

Collective	-	Vertical Velocity
Longitudinal Cyclic	-	Longitudinal Velocity
Lateral Cyclic	-	Lateral Velocity
Tail Rotor	-	Yaw Rate

Automatic Flight Control Systems (AFCS) are currently utilized in helicopters to reduce pilot workload. One of the most modern aircraft designs, the MH-60K, employs the following features to augment flying qualities in a hover:

1. Pitch, roll and yaw stability
2. Cyclic, collective and directional trim
3. Pitch and roll attitude hold
4. Heading hold
5. Altitude hold
6. Coupled hover (inputs through automatic pilot) [9]

These functions greatly enhance the mission effectiveness of the aircraft and crew in a combat aircraft employing many complicated weapons systems.

Flight Control Systems Design

AFCS for helicopters are primarily designed using classical or single input, single output (SISO) methods. These methods require that the multiple input, multiple output (MIMO) state space model of the system be broken down into scalar subsystems or transfer functions. Even recent research by Osder and Caldwell concludes that "A practical process for designing such multiple input, multiple output helicopter systems starts by decoupling controls into four single input, single output axes ... [10]." The simplification of the MIMO system is deemed necessary because SISO techniques become very complicated when all input-output relationships are addressed, especially cross coupling of controls and states. This normally leads control system designers to limit their focus to the worst cross coupling areas while ignoring the rest.

As an example, the UH-60 helicopter flight control system incorporates a mechanical mixing unit designed to eliminate coupled control inputs [11]. The mixing unit links only four out of the twelve possible cross couplings of the desired control-state matches. This is done even though coupling is present in all of the control-state pairings.

Modern design techniques allow for direct use of the MIMO state space model in control system design algorithms. There is no need to break down an integral model into SISO subsystems. Thus all the control state relationships, including cross couplings, are considered in the design process. Two popular MIMO design techniques are eigenstructure assignment and the Linear Quadratic Regulator.

Eigenstructure assignment allows control system designers to prescribe desired closed loop eigenvalues and eigenvectors for a given system, thus achieving desired performance characteristics. Garrard and Liebst used eigenstructure assignment "to design a feedback system for use in precise hovering control for a modern attack helicopter. Eigenvalue placement is used for stability enhancement and eigenvector shaping is used for modal decoupling [12]." But, as Moore has shown, eigenstructure assignment does not provide a unique control system design [13]. This flexibility allows the designer to augment eigenstructure assignment with additional design methods.

The Linear Quadratic Regulator (LQR) is an optimal control design method that yields a full state feedback controller which minimizes the quadratic performance index

$$J=\int_0^\infty (\mathbf{x}^T Q\mathbf{x} + \mathbf{u}^T R\mathbf{u})\,dt \qquad (2)$$

where Q is the symmetric, positive semi-definite state weighting matrix and R is the symmetric, positive definite control weighting matrix. Note that this performance index is quadratic in both state and control variations from nominal conditions, hence minimizing attempts to maintain small plant deviations with small control inputs. The LQR method can provide a robust closed loop solution with guaranteed stability margins [14:sect 6].

Using the eigenstructure assignment method, the control system designer can specify the desired performance of a given system. The LQR method provides a robust design solution. Combined use of these methods yields the best of both worlds.

Program Background

Captain Jeffrey D. Robinson developed an algorithm at AFIT which utilized eigenstructure assignment and the LQR method in defining linear, full state feedback gain for aerospace systems [1]. Robinson's algorithm was written exclusively for use with the MathWorks Inc. software package MATLAB™ [15] and was limited to eigenvalue assignment only. Captain Thomas C. Huckabone, also at AFIT, augmented Robinson's work by introducing eigenvector assignment [2]. Huckabone rewrote Robinson's program in FORTRAN using MATLAB™ only as a method of manipulating input and output. In addition to newly written routines, Huckabone's work utilized existing subroutines from the LQGLIB [16] and IMSL [17] packages available on the AFIT computer system.

The algorithm minimizes the performance index

$$J=\sum_{i=1}^{n}[Fe_i(\lambda_{d_i}-\lambda_{a_i})^2+(v_{d_i}-\theta_i v_{a_i})^* Fv_i(v_{d_i}-\theta_i v_{a_i})] \tag{3}$$

where

n	=	number of states
Fe	=	eigenvalue weighting matrix
λ_d	=	desired closed loop eigenvalue
λ_a	=	achieved closed loop eigenvalue
v_d	=	desired closed loop eigenvector
v_a	=	achieved closed loop eigenvector
Fv	=	eigenvector weighting matrix
θ	=	eigenvector minimization constant

The feedback controller is obtained via LQR methods, which minimize the LQR performance index presented in equation (2). Specifically, the algorithm varies the LQR performance index state and control weighting matrices, Q and R respectively, using an optimization method based upon the Nelder-Mead simplex algorithm presented in reference [18].

This thesis is a direct extension of Huckabone's work. His algorithm is modified so that the cross coupling weighting matrix S is included in the LQR performance index.

The performance index is now written as

$$J=\int_0^\infty \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} dt \qquad (4)$$

The modification was added to provide increased flexibility in achieving the desired eigenstructure. The algorithm is then applied to mathematical models of conventional, modern helicopters in hovering flight conditions. The eigenstructure and stability characteristics achieved via the modified algorithm are compared with results that do not utilize the cross coupling weighting matrix.

II. Cross Coupling Extension

Theory

The majority of the mathematical theory and equations necessary for development of the algorithm are reported in detail by Huckabone [2:9-22]. In particular, Garrard, Liebst [12] and Moore [13] provide eigenstructure assignment theory while Ridgely [14] provides LQR theory. Some general equations are presented below for convenience. The theory necessary to introduce the cross-coupling weighting matrix S is presented here in detail.

General. Again, the standard state equation of a multivariable, linear, time invariant, feedback system is

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{5}$$

Assuming full state feedback and a B matrix with full column rank yields a linear, feedback control law of

$$\boldsymbol{u} = -\boldsymbol{K}\boldsymbol{x} \tag{6}$$

Again, the standard LQR method involves minimizing the cost function

$$J = \int_0^\infty (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})\,dt \tag{7}$$

where Q is the symmetric, positive semi-definite state weighting matrix and R is the symmetric, positive definite control weighting matrix. From LQR theory, an optimal feedback gain matrix

$$K = R^{-1}B^{T}P \tag{8}$$

is obtained where the symmetric matrix P is the stabilizing solution to the algebraic Ricatti equation

$$PA + A^{T}P + Q - PBR^{-1}B^{T}P = 0 \tag{9}$$

<u>Cross-Coupling</u>. Introduction of the performance index

$$J=\int_{0}^{\infty}\begin{bmatrix}x\\u\end{bmatrix}^{T}\begin{bmatrix}Q & S\\S^{T} & R\end{bmatrix}\begin{bmatrix}x\\u\end{bmatrix}dt \tag{10}$$

allows for weighting and, consequently, minimization of combined state and control terms (i.e. $x_i u_j$) via the matrix S. The original LQR performance index in equation (7) allows minimization of pure state and control terms ($x_i x_j$ or $u_i u_j$) only[1]. The integrand of equation (10) can be expanded via matrix multiplication to

$$x^{T}Qx + x^{T}Su + u^{T}S^{T}x + u^{T}Ru \tag{11}$$

[1]The standard LQR performance index is equivalent to the new performance index when all elements of the matrix S are zero, hereafter referred to as S=[0].

Simplifying equation (11) via scalar addition and substituting back into equation (10) yields the rewritten performance index

$$J=\int_0^\infty (\mathbf{x}^T\mathbf{Q}\mathbf{x} + 2\mathbf{x}^T\mathbf{S}\mathbf{u} + \mathbf{u}^T\mathbf{R}\mathbf{u})\,dt \tag{12}$$

Anderson and Moore [19:56] show that standard LQR methods can be utilized to find an optimal feedback solution that minimizes a performance index written in the form of equation (12) with the following constraints[2]

$$\mathbf{R} > 0 \tag{13}$$

$$\mathbf{Q}-\mathbf{S}\mathbf{R}^{-1}\mathbf{S}^T \geq 0 \tag{14}$$

This is shown by rewriting the integrand of equation (12) as

$$\mathbf{x}^T(\mathbf{Q}-\mathbf{S}\mathbf{R}^{-1}\mathbf{S}^T)\mathbf{x} + \mu^T\mathbf{R}\mu \tag{15}$$

where

$$\mu = \mathbf{u} + \mathbf{R}^{-1}\mathbf{S}^T\mathbf{x} \tag{16}$$

Via substitution, the original state equation (5) is

[2]The expressions > 0 and ≥ 0, when used with matrices, refer to the matrix being positive definite and positive semi-definite, respectively.

rewritten as

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}^T)\boldsymbol{x} + \boldsymbol{B}\mu \tag{17}$$

The redefined LQR problem yields an optimal control law of

$$\mu = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{x} \tag{18}$$

where P is now the stabilizing solution to the algebraic Ricatti equation

$$\boldsymbol{P}(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}^T)+(\boldsymbol{A}^T-\boldsymbol{S}\boldsymbol{R}^{-1}\boldsymbol{B}^T)\boldsymbol{P}-\boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}+(\boldsymbol{Q}-\boldsymbol{S}\boldsymbol{R}^{-1}\boldsymbol{S}^T) = 0 \tag{19}$$

The optimal control law for the original system is then found by combining equations (16) and (18) which results in

$$\boldsymbol{u} = -\boldsymbol{R}^{-1}(\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{S}^T)\boldsymbol{x} \tag{20}$$

<u>Matrix Definiteness</u>. In order to apply the cross coupling theory, the algorithm must utilize a method that provides a standard LQR problem solver with inputs that satisfy the constraints of equations (13) and (14). This is done by creating the positive definite matrix

$$\bar{\boldsymbol{Q}} = \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^T & \boldsymbol{R} \end{bmatrix} \tag{21}$$

which satisfies the following necessary and sufficient condition for a positive definite matrix [20:331]

$$\mathbf{x}^T \mathbf{D} \mathbf{x} > 0 \quad \textit{for all vectors } \mathbf{x} \neq 0 \tag{22}$$

Defining

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \tag{23}$$

and substituting the partitioned matrix of equation (21) into the inequality (22) yields

$$\mathbf{x}_1^T \mathbf{Q} \mathbf{x}_1 + 2\mathbf{x}_1^T \mathbf{S} \mathbf{x}_2 + \mathbf{x}_2^T \mathbf{R} \mathbf{x}_2 > 0 \tag{24}$$

Setting $x_1=0$ yields

$$\mathbf{x}_2^T \mathbf{R} \mathbf{x}_2 > 0 \tag{25}$$

when $x_2 \neq 0$, which meets the necessary and sufficient condition for positive definiteness of matrix R, thus ensuring the constraint of equation (13) is met. Rewriting the inequality (24) as

$$\mathbf{x}_1^T (\mathbf{Q} - \mathbf{S}\mathbf{R}^{-1}\mathbf{S}^T) \mathbf{x}_1 + \chi^T \mathbf{R} \chi > 0 \tag{26}$$

where

$$\chi = \mathbf{x}_2 + \mathbf{R}^{-1}\mathbf{S}^T \mathbf{x}_1 \tag{27}$$

and setting $\chi=0$, $x_1\neq 0$ yields

$$x_1^T(Q-SR^{-1}S^T)x_1 > 0 \tag{28}$$

which meets the necessary and sufficient condition for positive definiteness of matrix $[Q-SR^{-1}S^T]$, thus satisfying the constraint of equation (14). A more general proof for satisfying these constraints is provided in Kreindler and Jameson [21:147].

Algorithm Changes

The original FORTRAN program EIGSPACE, as written by Huckabone, was changed where necessary to allow the use of the cross coupling weighting matrix S as described in the theory presented above. In addition, the program was cleaned up to eliminate unnecessary procedures. Details presented in this thesis represent only the major changes made for this thesis. The original EIGSPACE program and subroutine descriptions are presented in detail as appendices A and B of Huckabone's work [2:72-103]. The LQGLIB package of subroutines and the IMSL subroutine were not altered and are described in references [16] and [17], respectively. The modified program is presented in appendix C. Operating instructions and options are presented in appendix D.

The major changes to the original algorithm occur in two subroutines. Subroutine PP is modified to allow use of a standard LQR solver, in this case the LQGLIB subroutine REG. Subroutine MAKEQRS forms the weighting matrices Q, R and S. Also, the SORT subroutine was eliminated as it was unnecessary. Otherwise, the optimization methods and program flow for the algorithm are unchanged from that presented in detail in Huckabone's thesis [2:23-30].

Using the Standard Regulator (Subroutine PP). As stated in the cross coupling theory above, to modify a standard LQR solver to include the cross coupling weighting matrix S, the Q and A input matrices are redefined as

$$\boldsymbol{QNEW} = [\boldsymbol{Q}-\boldsymbol{S}\boldsymbol{R}^{-1}\boldsymbol{S}^{T}] \tag{29}$$

and

$$\boldsymbol{ANEW} = [\boldsymbol{A}-\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}^{T}] \ , \tag{30}$$

respectively. The regulator gain matrix of the redefined problem is then given as

$$\boldsymbol{G} = \boldsymbol{R}^{-1}\boldsymbol{B}^{T}\boldsymbol{P} \tag{31}$$

where P is the stabilizing solution to the Ricatti equation (19). The optimal control law for the desired problem is given in equation (20) and yields a regulator gain matrix for the desired system of

$$\boldsymbol{K} = \boldsymbol{R}^{-1}(\boldsymbol{B}^{T}\boldsymbol{P} + \boldsymbol{S}^{T}) \tag{32}$$

Thus the desired closed loop **A** matrix for the original system is

$$\mathbf{ACLS} = [\mathbf{A}-\mathbf{BK}] \qquad (33)$$

<u>Creating Q, R and S (Subroutine MAKEQRS)</u>. The parameters varied by the optimization subroutines are contained in the vector XGUESS. XGUESS makes up the upper triangular elements of the matrices QH and RM in the case where S is not varied. In the case where S is varied, XGUESS also contains all elements of the matrix SN. In both cases, the initial XGUESS is set such that Q and R are appropriately dimensioned identity matrices and, when appropriate, S=[0].

In the nonvariable S case, Q and R are formed as in the MAKEQR subroutine from the original program [2:100] with S=[0] introduced as input to the PP subroutine. The following description of how the algorithm creates the Q, R and S matrices applies only to the non-zero S case. While setting SN equal to an all zero matrix would properly form the Q and R matrices in the nonvariable S case, the algorithm requires that the two cases be handled differently. Specifically, in the non-variable S case, the XGUESS vector does not contain the additional parameters for SN as they would be unnecessarily iterated during the optimization process.

As stated in the matrix definiteness theory, in order to allow a variable S in the algorithm, the positive definite matrix

$$\bar{Q} = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \tag{34}$$

must be created. This matrix, renamed QRS in the program, can be formed as

$$QRS = MHN * MHN \tag{35}$$

where MHN is the real symmetric matrix

$$MHN = \begin{bmatrix} QN & SN \\ SN^T & RN \end{bmatrix} \tag{36}$$

Note that equation (35) is equivalent to

$$QRS = MHN^{*} MHN \tag{37}$$

where * denotes the Hermitian transpose of a matrix. Ridgely and Banda [14:2-7] provides that as long as **MHN** is nonsingular, the eigenvalues of QRS are all positive, hence QRS is positive definite [20:331]. The initial setting of **MHN** and the subsequent iteration process, virtually assure that **MHN** will always be nonsingular; however, the algorithm is designed to yield an error message if this occurs.

Substituting the partitioned MHM matrix of equation (36) into equation (35) yields:

$$QRS = \left[\begin{array}{c|c} QM*QM + SM*SM^T & QM*SM + SM*RM \\ \hline SM^T*QM + RM*SM^T & SM^T*SM + RM*RM \end{array}\right] \tag{38}$$

Thus, the weighting matrices generated by the algorithm are:

$$Q = QM*QM + SM*SM^T \tag{39}$$

$$R = RM*RM + SM^T*SM \tag{40}$$

$$S = QM*SM + SM*RM \tag{41}$$

<u>Eigenvalue Pairing (Subroutine SORT)</u>. As stated previously, the algorithm minimizes the performance index

$$J=\sum_{i=1}^{n} [Fe_i(\lambda_{d_i}-\lambda_{a_i})^2+(v_{d_i}-\theta_i v_{a_i})^* Fv_i(v_{d_i}-\theta_i v_{a_i})] \tag{42}$$

where

n	=	number of states
Fe	=	eigenvalue weighting matrix
λ_d	=	desired closed loop eigenvalue
λ_a	=	achieved closed loop eigenvalue
v_d	=	desired closed loop eigenvector
v_a	=	achieved closed loop eigenvector
Fv	=	eigenvector weighting matrix
θ	=	eigenvector minimization constant

The above performance index may not accurately reflect the designer's desired value during execution of the program. This is due to the fact that the algorithm must select which desired and achieved eigenvalues are to be paired together. (Note that the eigenvectors are paired in accordance with their respective eigenvalues.) Proper pairing can not be guaranteed by the program as it has no provisions for identifying eigenvalues by mode. Currently, the algorithm pairs the eigenvalues by comparing each achieved value with the value that is designated as the first desired eigenvalue and selecting the closest as its complement. This procedure continues for each desired eigenvalue, according to a selected order. The original EIGSPACE program attempted to utilize the SORT subroutine as a method of selecting the order. The subroutine sorted the eigenvalues in order of increasing magnitude and was used to sort both the desired and achieved eigenvalues.

Sorting the achieved eigenvalues was determined to be unnecessary since the algorithm ignores the sorted order when determining the closest match. Sorting the desired eigenvalues was also deemed unnecessary as the user may select the order when the desired eigenstructure is input to the program. It is important to note that the input order is important since different orders can yield different solutions. A simplified example illustrates this problem.

Figure 1 shows a possible mapping of achieved and desired eigenvalues. It appears obvious that in determining how close the achieved values are to the desired structure, a designer would pair the complex achieved eigenvalues with the complex desired eigenvalues and the real achieved value with the real desired value. However, if the real desired value is considered first in the pairing method used by the algorithm, it will pair off with one of the complex values, as they are closer. Thus the performance index will not accurately reflect what the designer would like to define as closeness. This will affect the minimizing path that the algorithm takes in searching for the best solution.

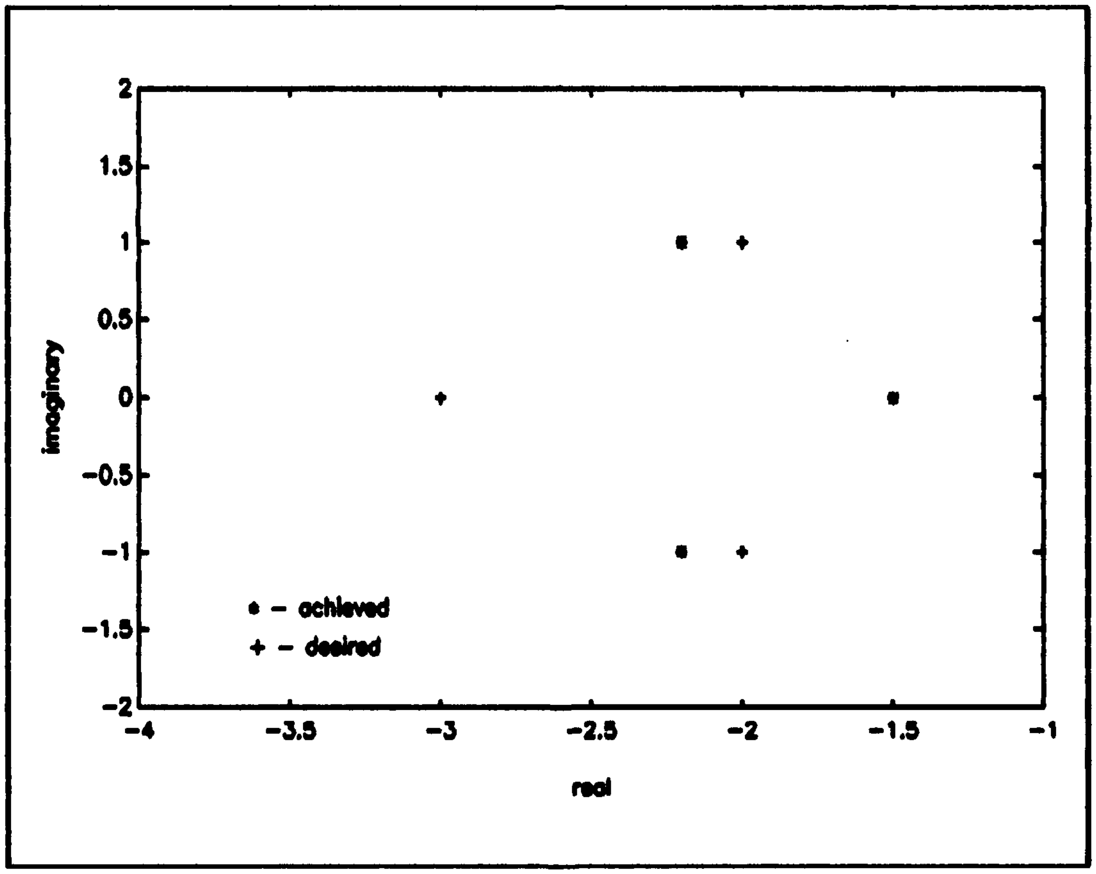

Figure 1
Example of Eigenvalue Pairing

It should also be noted that the input order of the sign of the imaginary component of desired complex eigenvalues is important as different pairings may occur for different orders. The algorithm is written so that the achieved eigenstructure determined by the DLQGLIB subroutine EIGVV always yields any complex pairs with the positive imaginary eigenvalue first. Knowing this, the user can input the desired eigenstructure accordingly.

For the simplified example above, the input order of the desired structure to avoid the mismatching problem is obvious. Unfortunately, higher state problems do not always provide the same easy insight and the number of permutations increases dramatically. Thus, the input order of the desired eigenstructure becomes another designer chosen parameter.

III. Stability Robustness

Theory

Ridgely and Banda [14] present theory for stability robustness of MIMO systems as well as guaranteed stability margins using the LQR solution. Specifically, the notion of independent gain and phase margins is introduced as follows:

> Independent gain margins (IGM) are limits within which the gains of all feedback loops may vary independently at the same time without destabilizing the system, while the phase angles remain at their nominal values. Independent phase margins (IPM) are limits within which the phase angles of all loops may vary independently at the same time without destabilizing the system, while gains remain at their nominal values [14:3-73].

The following relationships are shown to exist

$$\frac{1}{1+\alpha} < IGM < \frac{1}{1-\alpha} \tag{43}$$

and

$$-\sin^{-1}\left(\frac{\alpha}{2}\right) < \frac{IPM}{2} < \sin^{-1}\left(\frac{\alpha}{2}\right) \tag{44}$$

where α is the minimum singular value, for all frequencies, of the return difference matrix given by

$$\alpha = \inf_{\omega} \underline{\sigma}\,[I+K(j\omega I-A)^{-1}B] \tag{45}$$

Note that equation (45) must satisfy the constraint $\alpha \leq 1$.

The inequalities (43) and (44) are conservative, thus they yield guaranteed minimum stability margins.

Ridgely and Banda [14:sect 7] also derive the Kalman inequality from LQR relationships, which is:

$$[I+R^{\frac{1}{2}}K(j\omega I-A)^{-1}BR^{-\frac{1}{2}}]^T[I+R^{\frac{1}{2}}K(j\omega I-A)^{-1}BR^{-\frac{1}{2}}] \geq I \qquad (46)$$

Guaranteed LQR stability margins are derived from equation (46) under the restriction $R = \rho I$, where ρ is any positive scalar. Hence, the Kalman inequality (46) simplifies to

$$[I+K(j\omega I-A)^{-1}B]^T[I+K(j\omega I-A)^{-1}B] \geq I \qquad (47)$$

This can be true only when

$$\alpha = \underline{\sigma}[I+K(j\omega I-A)^{-1}B] \geq 1 \qquad (48)$$

Substituting $\alpha = 1$ into equations (43) and (44) yields the following guaranteed minimum LQR stability margins under the restriction $R = \rho I$.

$$\frac{1}{2} < IGM < \infty \qquad (49)$$

$$-60^\circ < IPM < 60^\circ \qquad (50)$$

<u>Achievable Robustness with Cross Coupling Weights</u>

As reported by Huckabone [2], Safonov and Athans [22] show that when R is diagonal, the stability margins of equations (49) and (50) hold as long as the perturbations to each channel of the system occur independently. Independent perturbations are implied by an R matrix having elements of the same relative magnitudes. Huckabone points out that:

> For the case of any general R, the independent stability margins ... cannot be guaranteed and often will go outside of these bounds. However, as previously mentioned, the equations for IGM and IPM provide conservative values. While the choice of any general R may not provide the guaranteed stability margins ... the system may still provide acceptable stability characteristics [2:22].

Introduction of the S matrix in the algorithm virtually assures that R will not be diagonal. This is seen by reviewing the equation

$$R = RM*RM + SN^T*SN \tag{51}$$

In the algorithm, the positive definite matrix RM*RM can be restricted via input codes as follows:

r-code = 1 $\rightarrow$ RM*RM = ρI

r-code = 2 $\rightarrow$ RM*RM is diagonal

r-code = 3 $\rightarrow$ RM*RM is general

In the cases where r-code = 1 or 2, the off-diagonal elements of R can only be zero if SN^T*SN is diagonal. This would require relationships between elements of SN. Since each element of SN is independently generated, no forced correlations between elements exist. While possible, it is highly unlikely that the elements of SN will randomly meet the requirements for SN^T*SN to be diagonal. In the case where r-code = 3, correlations between SN and RM must be met; therefore, it is even more unlikely that R can be diagonal.

Fortunately, it is possible that the algorithm will provide an R matrix close to diagonal. Huckabone's results for the AH-64 helicopter yield an R matrix close to diagonal.

> The significance of this R matrix is that because it comes close to approximating a diagonal matrix, the minimum singular value of the return difference matrix is nearly one. It turns out that R being near diagonal is a general result for all of the cases run with this example for R > 0. Therefore ... the resulting closed loop systems will still possess good independent stability margins [2:59].

Again looking at equation (51), If RM*RM is close to diagonal, the resulting R will be close to diagonal if

$$|[RM*RM]_{ij}| \gg |[SN^T*SN]_{ij}| \tag{52}$$

While the preceding shows that good stability margins are possible, it also demonstrates that stability margins are no longer guaranteed. In fact, it can be shown that the eigenvalues of a system can be placed anywhere within the stable region of the complex plane using cross coupling weights.

Robinson [1] demonstrated how his version of the algorithm would find solutions only within an achievable LQR region for a two state, single input system defined by the following matrices:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Recall that Robinson's algorithm did not use cross coupling weights and matched eigenvalues only. Note that for a single input system, $R = \rho I$ is satisfied, thus the LQR guaranteed stability margins of equations (49) and (50) apply. In fact these margins restrict the solution, thus preventing the algorithm from achieving the desired eigenvalues. The algorithm developed in this thesis, using cross coupling weights, was able to achieve eigenvalues outside of that restricted region. In fact, using the system described by the above A and B matrices, it can be demonstrated that the LQR with cross coupling weights can achieve any closed loop stable solution, thus inferring that there are no guaranteed stability margins when cross coupling weights are applied in the algorithm.

The closed loop characteristic equation for a system is defined as:

$$det[\lambda \boldsymbol{I} - \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}] = 0 \tag{53}$$

For the system defined above by A and B, with K defined as:

$$\boldsymbol{K} = [k_1 \ k_2]$$

equation (53) becomes:

$$\lambda^2 + k_2\lambda + k_1 = 0 \qquad (54)$$

Thus the ability to arbitrarily assign $k_1 > 0$ and $k_2 > 0$ would provide closed loop eigenvalue assignability within the entire stable (or left) half of the complex plane.

For this example, let the LQR weighting matrices be defined as:

$$\boldsymbol{Q} = \begin{bmatrix} q_{11} & 0 \\ 0 & q_{22} \end{bmatrix} \qquad \boldsymbol{R} = 1 \qquad \boldsymbol{S} = \begin{bmatrix} s_1 \\ 0 \end{bmatrix}$$

The regulator gain and Ricatti equations using cross coupling weights are repeated here for convenience:

$$\boldsymbol{K} = \boldsymbol{R}^{-1}(\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{S}^T) \qquad (55)$$

$$\boldsymbol{P}(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{S}^T) + (\boldsymbol{A}^T-\boldsymbol{S}\boldsymbol{R}^{-1}\boldsymbol{B}^T)\boldsymbol{P}-\boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + (\boldsymbol{Q}-\boldsymbol{S}\boldsymbol{R}^{-1}\boldsymbol{S}^T) = 0 \qquad (56)$$

where P for this example can be defined as follows:

$$\boldsymbol{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}$$

Solving for the regulator gain in equation (55) for this example yields:

$$k_1 = p_{12} + s_1 \tag{57}$$

$$k_2 = p_{22} \tag{58}$$

Expanding the Ricatti equation (56) yields the following relationships:

$$(p_{12} + s_1)^2 = q_{11} \tag{59}$$

$$(p_{22})^2 = q_{22} + 2p_{12} \tag{60}$$

Combining these results yields the following equations for the scalar gains in terms of the LQR weights:

$$k_1 = (q_{11})^{1/2} \tag{61}$$

$$k_2 = [q_{22} + 2((q_{11})^{1/2} - s_1)]^{1/2} \tag{62}$$

As was shown earlier, requiring the matrix

$$\begin{bmatrix} Q & S \\ S^T & R \end{bmatrix}$$

to be positive definite satisfies the necessary constraints for Ricatti equation (56) to provide a stabilizing solution. In this example, positive definiteness of the above matrix provides the following constraints:

$$q_{11} > 0$$

$$q_{22} > 0$$

$$q_{11} > s_1^2$$

These constraints and equation (61) show that k_1 can be selected arbitrarily such that $k_1 > 0$. Also from the constraints, s_1 can be selected such that

$$(q_{11})^{1/2} - s_1$$

approaches zero; therefore, from equation (62), k_2 is solely dependent on q_{22} and can also be selected arbitrarily to satisfy $k_2 > 0$. As stated before, this allows for assigning any closed loop eigenvalues in the left half of the complex plane.

Recall that the regulator gains and Ricatti equations given above are extensions of standard LQR theory and are valid without cross coupling weights. To revert back to the standard LQR relationships, S=[0] is substituted into the appropriate equations. For this example, forcing S=[0] yields the following:

$$k_2 = [q_{22} + 2(q_{11})^{1/2}]^{1/2} \qquad (63)$$

Notice from equation (61) that k_1 can still be arbitrarily selected via q_{11}, but k_2 is now restricted by the selection of k_1. In fact, k_2 can never be less than $(2k_1)^1$ and hence the LQR achievable region is restricted to a closed loop damping factor of greater than 0.707. Thus it is easily seen that the addition of the cross coupling weight, s_1 in this two state example, directly allows for the arbitrary placement of the closed loop eigenvalues within the left half complex plane. Therefore, the restrictions imposed by the standard LQR solution, without cross coupling, are removed. Unfortunately, the minimum gain and phase margins of equations (49) and (50) are now no longer guaranteed.

IV. Results

Inputs

Within this thesis, numerical data has been rounded off for ease of presentation, with a goal of accuracy to the fourth significant digit. Input data must be accurate to the eighth significant digit in order to precisely duplicate the results presented here. Differences between results in this thesis with those in Huckabone's work [2] using the AH-64 model are directly attributable to a difference in accuracy beyond the fourth significant digit. The AH-64 model, represented in appendix A, is displayed exactly as input to the algorithm in this thesis. Inputs to the algorithm include:

A, B	- matrices from models
Ed	- diagonal matrix containing the desired eigenvalues
Vd	- modal matrix of desired eigenvectors (columns correspond to those of Ed)
Fe	- row vector of weights corresponding to the diagonal elements of Ed
Fv	- matrix of weights corresponding to vd
tol	- convergence tolerance
r-code	- 1 $\Rightarrow$ $RM^*RM = \rho I$ 2 $\Rightarrow$ RM^*RM is diagonal 3 $\Rightarrow$ RM^*RM is general
s-code	- 0 $\Rightarrow$ $S = [0]$ 1 $\Rightarrow$ S is filled
kmax	- maximum optimization iterations

The desired closed loop eigenstructure for both models is the same as developed by Garrard and Liebst [12] and is presented in Table I. The selected eigenvalues and eigenvectors were based on work by Hoh [23], which was a precursor to ADS-33 [8]. Unity values represent desired state-to-state coupling. The non-unity elements corresponding to ϕ and θ are the inverses of the desired roll and pitch eigenvalues, as these states are the integrals of the respective rates. An X denotes an element of the eigenvector where coupling is inevitable. These values are allowed to float freely in the algorithm by applying a zero weighting factor to the corresponding element in the matrix Fv.

Table I
Desired Closed Loop Eigenstructure

State	Mode					
	Long Vel	Lat Vel	Heave	Pitch	Yaw	Roll
λ	-.801 ± .387i	-.802 ± .388i	-1.0	-2.9	-3.0	-3.5
u	1	0	0	x	0	0
v	0	1	0	0	0	x
w	0	0	1	0	0	0
p	0	x	0	0	0	1
q	x	0	0	1	0	0
r	0	0	0	0	1	0
ϕ	0	x	0	0	0	-.2857
θ	x	0	0	-.345	0	0

A small problem with the algorithm was discovered during the research to determine stability margins. When desired eigenvalues were input as unstable, the algorithm did not work properly. Specifically, the algorithm yielded an unstable solution. It is felt that this result is due to a fault within the DLQGLIB subroutine REG which allows for an unstable solution. If an unstable solution is provided as a possible achieved solution to be compared to an unstable desired structure, the algorithm may select it as the solution that minimizes $\mathcal{J}$. In this thesis, unstable desired structures were input only in an attempt to determine if any achievable region could be mapped out for the LQR with cross coupling weights. As there are few instances where an unstable solution is desired, this problem does not appear to provide any major obstacles in the effective use of the algorithm.

As was mentioned before, the order is important in entering the desired eigenstructure. Figure 2 shows the desired eigenvalues and achieved eigenvalues for the AH-64 where Q and R are appropriately dimensioned identity matrices, which is the first guess in the algorithm. Some insight is gained from this in deciding the input order. Specifically, the complex desired eigenvalues must be paired before the real eigenvalues in order to ensure that they are matched up with complex achieved eigenvalues. Also, the desired eigenvalue at (-3.5, 0) must be paired last so that it will match up with the achieved eigenvalue at (-18.1, 0).

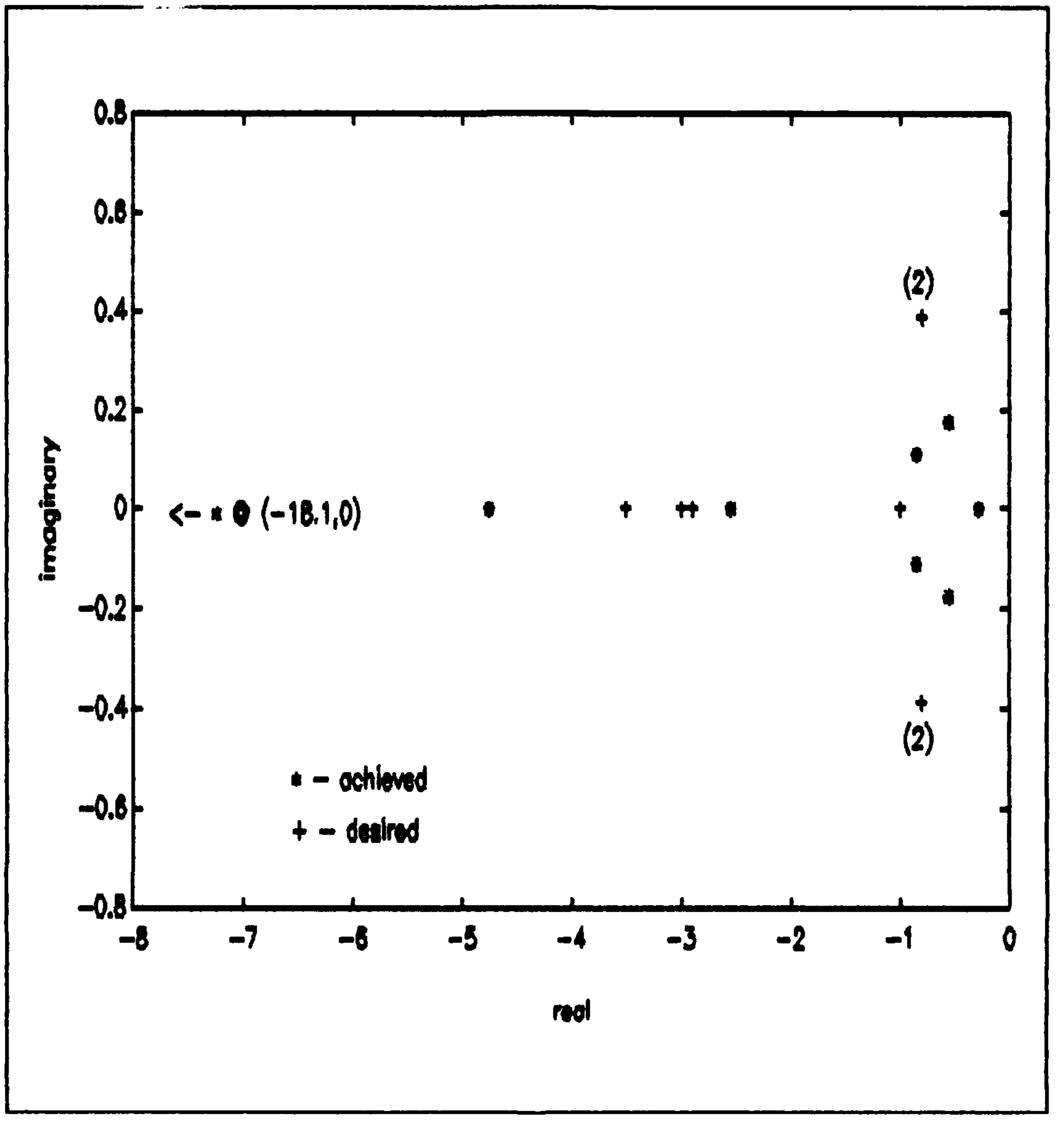

Figure 2
AH-64 Eigenvalue Pairing

Thus the desired helicopter eigenstructure for this thesis was always input with the most dominant eigenvalue first proceeding to the least dominant, unless otherwise stated. The eigenvalue with the positive imaginary component is always input first.

Unity weighting of the eigenvalues refers to each element of Fe being set equal to one. Unity weighting of the eigenvectors refers to each element of Fv being set equal to one except those elements corresponding to the components of Vd that are free to float (see Table I). These elements are always set equal to zero.

The convergence tolerance (tol) is used in comparing $\mathcal{J}$ for consecutive iterations to determine if a good minimum has been achieved. It was discovered that the program occasionally reaches plateaus where very small improvements to $\mathcal{J}$ are made which can be followed by much larger improvements. This is illustrated in Table II. Too high of a convergence tolerance will prevent the program from achieving these large improvements. Too low of a tolerance may not allow the program a natural stopping point, thus a maximum iteration value, kmax, must be set.

Unless otherwise stated, a value of 10^{-4} is used for tol and 150 is used for kmax. While 10^{-4} may be beyond the accuracy desired in measuring the performance index, there were cases where higher values for tol were shown to mask a 27% reduction of $\mathcal{J}$. And, while a kmax value of 150 would appear to yield adequate opportunity for the program to converge, cases were run to this limit utilizing over eight hours of central processing unit time, while continuing to reduce $\mathcal{J}$ by more than 10^{-3}. It is felt, however, that these values provide a good basis for comparison of results using different r-codes and s-codes.

Table II
Sample $\mathfrak{J}$ Convergence

Iteration	$\mathfrak{J}$	Improvement
1	10.961	-
2	9.523	1.438
3	7.703	1.820
4	7.392	0.311
5	7.134	0.258
6	6.943	0.191
7	6.829	0.114
8	6.766	0.063
9	6.533	0.233
10	6.447	0.086
11	6.405	0.042
12	6.272	0.133
13	6.224	0.048
14	6.153	0.071
15	6.039	0.114
16	5.985	0.054
17	5.943	0.042
18	5.577	0.366
19	5.130	0.447
20	4.662	0.468

The r-codes and s-codes are specified for each example. Recall that for the case where S is allowed to vary, s-code = 1, the R matrix can only be restricted to positive definite.

Analysis of Results

The algorithm's eigenstructure performance index, $\mathfrak{J}$, presented in equation (42) can be broken down into two elements depicting the contributions of the eigenvalue and eigenvector differences. Unfortunately, for weighting values other than unity, a true measure of the distance from the achieved to desired eigenstructure will not be reflected in the performance index or its parts. In addition, the achieved and desired eigenvalues may not be properly paired by the algorithm, as was previously discussed.

In order to analyze the performance of the algorithm solutions, the parameter J_λ is introduced as

$$J_\lambda = \sum_{i=1}^{n} (\lambda_{d_i} - \lambda_{a_i})^2 \tag{64}$$

where the parameters are paired by mode. This gives a true measure of the closeness of the achieved solution's eigenvalues to the desired eigenvalues. Note that when unity weighting of the eigenvalues is used and the returned solution does properly match eigenvalues, J_λ does reflect the eigenvalue contribution to $\mathfrak{J}$.

Modal decoupling is analyzed via the eigenvectors. The algorithm normalizes all eigenvectors in the program, including the output. The achieved eigenvectors presented here have been multiplied by the inverse of the eigenvector element corresponding to the primary desired response element of the desired eigenstructure. These are the elements valued at unity in Table I. Thus the non-unity elements of each eigenvector may be viewed as a distance away from zero coupling of the respective element.[3]

[3]The free-to-float and non-unity elements of the desired eigenstructure are annotated in the data presentation tables

AH-64 Results

Achieving a Closer Solution. Table III shows a summary of the results for the AH-64 model using unity weighting.

Table III
Summary of AH-64 Results (Unity Weighting)

Run #	r-code	s-code	$\bar{J}$	α
1	1	0	4.240	1
2	2	0	3.560	.89371
3	3	0	3.333	.47105
4	1	1	6.219	.88463
5	2	1	11.503	.48946
6	3	1	2.847	.80189

The first three runs of this example show expected results, in that reduced restrictions on R allow for more flexibility in reducing $\bar{J}$. Note that the addition of the cross coupling weights in run 6 results in a 17% decrease of $\bar{J}$. Runs 4 and 5 demonstrate a problem with the algorithm. Recall that when using a filled S matrix, s-code = 1, the r-code selected can not directly effect the form of the R matrix as it does with an S=[0], s-code = 0. Thus the r-codes only effect the path taken by the algorithm in finding an optimal solution. Additionally, the algorithm does not provide for the possibility of following the same path as with s-code = 0, which in this case would provide a better solution.

A surprising result from run 3 is relatively poor robustness. The IGM and IPM associated with run 3 are [0.680, 1.890] and [-27.2°, 27.2°], respectively. These margins indicate that the algorithm will not always produce an R matrix that yields good robustness properties, even without the use of cross coupling weights. This problem is addressed later in the thesis. Run 6 provides acceptable IGM and IPM of [0.555, 5.047] and [-47.3°, 47.3°], respectively. Minimum singular values for runs 1, 3 and 6 are presented in Figure 3 for comparison.

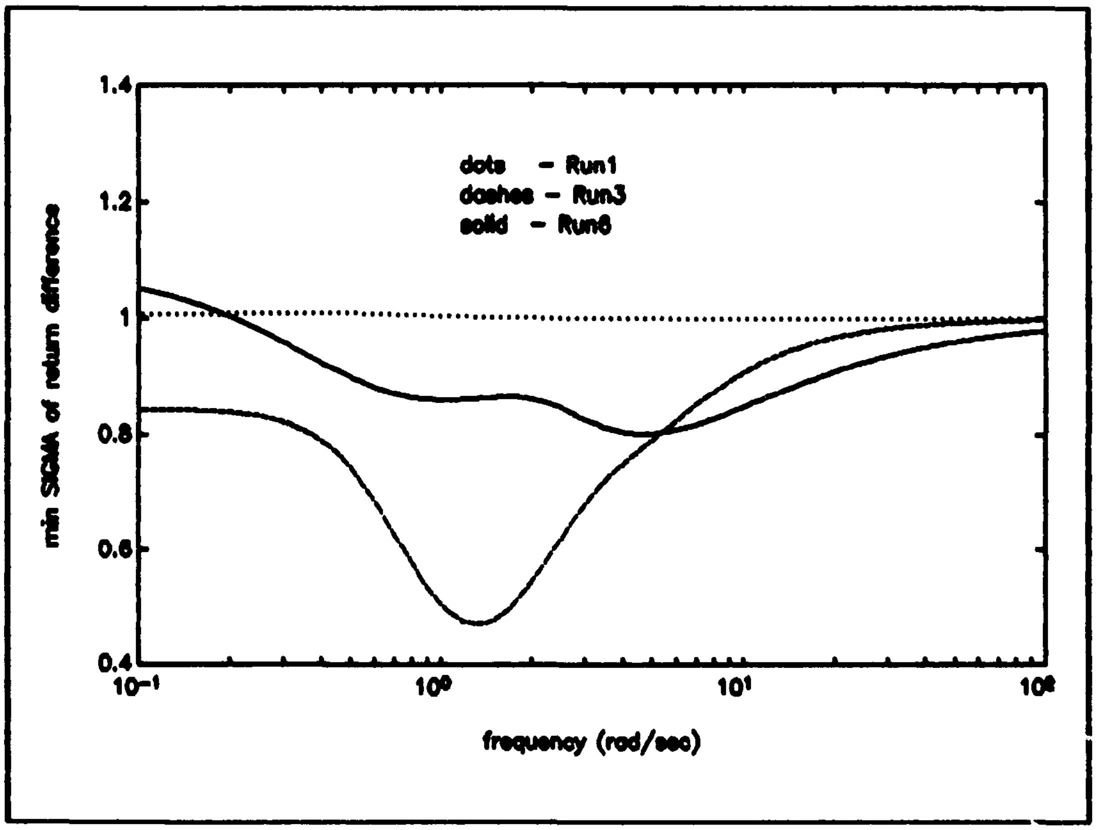

Figure 3
AH-64 Minimum Singular Values

The achieved eigenstructure for runs 3 and 6 are presented in Table IV and Table V, respectively. The appropriate gain matrix, K, is presented below each table.

Table IV
AH-64 Achieved Closed Loop Eigenstructure (Run 3)

State	Mode					
	Long Vel	Lat Vel	Heave	Pitch	Yaw	Roll
λ	-0.592± 0.233i	-0.551± 0.201i	-0.978	-2.853	-2.821	-3.691
u	1	0.089± 0.053i	2.100	-.136*	0.064	-0.001
v	-0.078∓ 0.083i	1	5.351	-0.028	-0.038	0.043*
w	0.208± 0.001i	0.032∓ 0.068i	1	-0.053	-0.092	-0.044
p	0.056∓ 0.018i	0.345∓ 0.325i*	7.282	0.097	0.093	1
q	-0.650± 0.528i*	0.046∓ 0.019i	-3.028	1	-0.012	0.214
r	0.052∓ 0.143i	0.061∓ 0.029i	0.766	-0.026	1	0.706
φ	-0.115± 0.015i	-0.754± 0.320i*	-7.542	-0.031	-0.070	-0.290^{r}
θ	1.246± 0.389i*	-0.088± 0.005i	3.054	-.350^{p}	-0.014	-0.067

Notes: * denotes value free to float
^{p} denotes desired value of -0.345
^{r} denotes desired value of -0.286

```
K =  Columns 1 through 4

   4.4662e-01   2.0166e-01  -1.0815e+00  -2.6179e-01
   3.1036e-01  -2.2492e-01   4.9597e-01  -8.6887e-02
   3.7330e-02   1.7525e-02   8.8220e-01   9.8005e-02
  -1.6387e-01  -2.5946e-02   6.3103e-01   1.9216e-02

     Columns 5 through 8

   5.6113e-01   1.2189e+00  -8.2418e-02   5.7191e-02
  -7.0436e-01  -5.8664e-02  -1.7295e-01  -8.3843e-01
   7.3459e-02   2.9645e-01   5.0998e-01  -6.8360e-02
   3.9136e-01  -4.9041e-01  -1.4568e-01  -2.2571e-01
```

Table V
AH-64 Achieved Closed Loop Eigenstructure (Run 6)

State	Mode					
	Long Vel	Lat Vel	Heave	Pitch	Yaw	Roll
λ	-0.731± 0.334i	-0.823± 0.302i	-0.281	-2.941	-2.859	-3.636
u	1	0.065± 0.047i	-0.046	-.107*	-0.012	0.034
v	-0.094∓ 0.075i	1	-0.037	-0.018	0.114	0.078*
w	0.220± 0.016i	0.108∓ 0.032i	1	-0.059	-0.057	-0.007
p	0.039∓ 0.023i	0.807∓ 0.755i*	-0.009	-0.144	1.019	1
q	-0.950± 0.897i*	0.035∓ 0.005i	-0.017	1	0.393	-0.224
r	-0.058∓ 0.214i	-0.053± 0.063i	-0.049	0.196	1	0.597
ϕ	-0.069± 0.037i	-1.153± 0.486i*	0.048	0.044	-0.392	-0.292^{R}
θ	1.534∓ 0.510i*	-0.035∓ 0.011i	0.069	-.343^{P}	-0.155	0.053

<u>Notes</u>: * denotes value free to float
^{P} denotes desired value of -0.345
^{R} denotes desired value of -0.286

```
K =   Columns 1 through 4

      -2.3441e-01   5.4556e-01  -1.1410e-01   8.6480e-03
       7.4176e-01  -2.3506e-01   8.1201e-02   2.0316e-03
       1.0283e-01   1.1148e-02   3.5349e-03   2.4469e-01
      -7.7074e-02   3.3675e-01   7.8341e-02  -2.1966e-02

      Columns 5 through 8

       6.1181e-01   6.4257e-01   3.8626e-01   5.4422e-01
      -6.7483e-01  -5.1553e-02  -1.4942e-01  -1.0433e+00
      -7.6641e-02  -1.0873e-01   5.2847e-01  -6.8685e-02
       2.3170e-01   1.4014e-01   3.1558e-02   1.7497e-01
```

While the addition of the cross coupling weighting in run 6 does achieve a desired reduction of $\mathfrak{J}$, the eigenvalue portion, J_λ, increases from 0.365 to 0.588. This increase is due to the placement of the heave eigenvalue in run 6, which accounts for 88% of J_λ. In fact, all other eigenvalues are closer to desired in run 6 than in run 3. The improvements in the overall $\mathfrak{J}$ lie in the eigenvectors, particularly the heave associated eigenvector. Coupling is tremendous in run 3; in fact, vertical velocity (w) is not the primary response element, as is desired.

As mentioned earlier, a precise comparison between the above results and Huckabone's [2] results for the same model was not possible; however, some important differences were discovered. Of most importance for this thesis is the fact that the addition of cross coupling weights did allow for a solution closer to the desired eigenstructure than any previous solutions without cross coupling weights, where closeness is measured via $\mathfrak{J}$. Also notable was the reduced stability robustness obtained without the addition of cross coupling weights in run 3.

Convergence Tolerance Effects on Stability Margins. The poor stability margins for run 3 were somewhat surprising in that they were much worse than those previously obtained in Huckabone's work [2]. While it has previously been shown that the addition of cross coupling weights to the algorithm may yield poor robustness, run 3 did not include this addition. Further research revealed that the convergence tolerance input to the program was responsible for relaxed stability margins.

Recall that the LQR method of control system design is utilized primarily to take advantage of good robustness properties. These properties were shown to be dependent on the type of R matrix used within LQR theory. As stated before, an R matrix close to diagonal could be expected to produce good stability margins. The algorithm begins its search for an optimal solution by perturbing away from R=I, which provides the guaranteed stability margins presented in equations (49) and (50). As R is perturbed away from I, or more correctly away from ρI, R becomes far from diagonal and the stability margins become worse.

In searching for the lowest possible $\mathfrak{J}$, the convergence tolerance value was decreased significantly below the values used by Huckabone [2]. As mentioned above, this was done to take advantage of the apparently unending improvements that were gained with the introduction of cross coupling weighting and to provide for fair comparison of results. What it also did, was allow R to perturb much farther away from diagonal than it had been previously allowed, thus revealing poor stability robustness characteristics.

In order to demonstrate this problem, run 3 was repeated with an increased tol value of 10^{-2} versus the previous 10^{-4}. The solution yielded $\mathfrak{J}$ = 4.67 and

$\alpha = 0.9774$. $\mathcal{J}$ is slightly higher, but a significant improvement is seen versus the previous $\alpha = 0.4715$. The R matrix of run 3, where tol is 10^{-4}, is returned as

$$\mathbf{R} = \begin{bmatrix} 2.0325 & 0.4111 & 0.5768 & 0.9062 \\ 0.4111 & 2.6846 & -0.0203 & -0.2034 \\ 0.5768 & -0.0203 & 3.3425 & 0.7791 \\ 0.9062 & -0.2034 & 0.7791 & 3.3992 \end{bmatrix}$$

The R matrix where tol is 10^{-2} is

$$\mathbf{R} = \begin{bmatrix} 2.0971 & 0.6235 & 0.3868 & 0.4499 \\ 0.6235 & 2.2980 & 0.0294 & 0.0437 \\ 0.3868 & 0.0294 & 2.6814 & 0.6522 \\ 0.4499 & 0.0437 & 0.6522 & 2.1886 \end{bmatrix}$$

Notice that for the lower tolerance run, R has a larger spread between diagonal elements as well as larger off diagonal elements, which is what is meant by being farther away from diagonal.

The loss of stability margins can be even more pronounced when cross coupling weighting is introduced. While the above logic of perturbing R away from I still holds, recall that R can now be perturbed by twice as many variables, and may thus be perturbed away twice as fast. This is seen by reviewing the equation

$$\mathbf{R} = \mathbf{RM}*\mathbf{RM} + \mathbf{SM}^T*\mathbf{SM} \qquad (65)$$

Nevertheless, the stability margins shown above for run 6 are not completely inadequate. In fact, stability is not a requirement for the hovering helicopter applications presented in this thesis according to current handling qualities requirements [8:20]. And, when run 6 is repeated with a higher convergence tolerance of 10^{-8}, α is improved from 0.8019 to 0.8612 with a resulting $\mathfrak{J}$ = 3.204. This $\mathfrak{J}$ is still lower than any previous runs for this model without cross coupling weighting.

UH-60 Results

Dimensional Effects. The UH-60 model was input to the algorithm with unity weighting on the eigenstructure. A summary of results is presented in Table VI. The algorithm was unable to provide close matches to the desired eigenstructure. Also, introduction of the cross coupling weighting did not improve the algorithm's performance. Recall that, unlike the AH-64 model, the UH-60 model is dimensional. This prevents the algorithm from providing adequate results.

Table VI
Summary of UH-60 Results (Unity Weighting)

Run #	r-code	s-code	$\mathfrak{J}$
7	1	0	12.876
8	2	0	6.693
9	3	0	7.483
10	1	1	8.113
11	2	1	19.707
12	3	1	9.159

Because of the absence of dimensional conditioning of the system, the differences in units of measure forces the algorithm to select weighting values that make up for the differences in dimensions. This can be seen by reviewing the resulting R matrix for the best solution, run 8:

$$\mathbf{R} = \begin{bmatrix} 8.040 & 0 & 0 & 0 \\ 0 & 0.826 & 0 & 0 \\ 0 & 0 & 2.448 & 0 \\ 0 & 0 & 0 & 0.056 \end{bmatrix}$$

As would be expected from the above matrix, robustness for this solution is poor as demonstrated by $\alpha = 0.523$. The requirement to vary the weighting matrices to account for dimensions effectively constrains the path that the algorithm must take in finding an optimal solution. Thus the algorithm reaches a dead end that it would normally avoid without the dimensionally imposed constraints. To avoid this problem, the UH-60 model was non-dimensionalized. Changing scales of dimensions (i.e. ft/sec to m/sec) could be used to provide the same affect.

The following maximum values were used for the given states and inputs in non-dimensionalizing the UH-60 model:

60 ft/sec	- u, v, w
60 °/sec	- p, q, r
45 °	- ϕ, θ
8 inches	- δ_a, δ_b, δ_c
4 inches	- δ_p

For the non-dimensional UH-60, with unity weighting, the algorithm returned a solution with $\mathfrak{J}$ = 4.145 and α = 0.765 with r-code = 3 and s-code = 1. Eigenvalue matching was very good with J_λ = 0.185. The achieved eigenvector associated with the heave mode is

$$V_{a_{heave}} = \begin{bmatrix} 0.2014 \\ -1.7830 \\ 1.0000 \\ -3.2322 \\ -0.4236 \\ 0.0643 \\ 3.2208 \\ 0.4188 \end{bmatrix}$$

This is obviously a poor match accounting for over 40% of the eigenvector contribution to $\mathfrak{J}$. Using these results as a baseline, several variations were run to illustrate some designer options.

<u>Eigenstructure Input Order Effects</u>. First, a unity weighting case for the non-dimensional UH-60 model was run with a different desired eigenstructure order to demonstrate the affect on the optimization path. The new eigenvalue order was arbitrarily selected as:

1. -0.801 - 0.387i
2. -0.801 + 0.387i
3. -0.802 - 0.388i
4. -0.802 + 0.388i
5. -1.0
6. -2.9
7. -3.5
8. -3.0

This run yielded the following results: $\mathfrak{J} = 3.575$, $J_\lambda = 0.167$ and $\alpha = 0.754$. This represents a 14% decrease in $\mathfrak{J}$ with only a slight reduction in stability margins. Obviously, as the program approaches a solution where the achieved eigenvalues are close to the desired values, as is the case for this and the baseline runs, the eigenvalue pairing will be correct. But, as was shown before, the initial solutions may yield eigenvalues far from desired, resulting in different pairings. This initial pairing affects the path that the algorithm follows in searching for an optimal solution. For the initial run with the non-dimensional UH-60 model, the solution was achieved after 7 optimization iterations. The run with the new order completed 67 iterations before yielding a solution. Note that in both cases, the standard convergence tolerance of 10^{-4} was used.

Weighting Element Changes. The next change to the baseline run was a single element weighting change. Since, the eigenvector associated with the heave mode was poorly decoupled in the initial run, the weight on the vertical velocity component (w) was increased from 1 to 4. The resulting solution yielded
$\mathfrak{J} = 2.815$, $J_\lambda = 0.171$ and $\alpha = 0.799$. The emphasis on the heave eigenvector, through the increased weight on the desired response element, allowed the algorithm to find a closer solution with better stability margins. The heave

eigenvector was dramatically decoupled as seen here

$$V_{a_{heave}} = \begin{bmatrix} -0.0353 \\ -0.0299 \\ 1.0000 \\ 0.0050 \\ -0.0057 \\ 0.1083 \\ -0.1228 \\ 0.0002 \end{bmatrix}$$

In order to demonstrate the potential of the program, several iterations of the non-dimensional UH-60 model were run. The following designer weighting matrices were used:

$$\mathbf{Fe} = [\ 2 \quad 2 \quad 2 \quad 2 \quad 3 \quad 2 \quad 2 \quad 2\ ]$$

$$\mathbf{Fv} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1.5 & 1.5 & 3 & 3 & 6 & 2 & 2 & 1 \\ 1 & 1 & 0 & 0 & 1 & 3 & 4 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 5 & 2.5 \\ 1 & 1 & 2 & 2 & 1 & 4 & 6 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 3 & 1 \end{bmatrix}$$

The convergence tolerance was set at 10^{-8} in an effort to obtain good stability margins. The algorithm achieved a $\mathcal{J}$ of 2.276. Note that the algorithm achieved a closer match than previous iterations even with non-unity weights included in $\mathcal{J}$. The minimum singular value, $\alpha = 0.808$, yields an IPM of [47.7°, -47.7°] and an IGM of [0.553, 5.214]. The achieved closed loop eigenstructure is given in Table VII.

Table VII
UH-60 Achieved Closed Loop Eigenstructure

State	Mode					
	Long Vel	Lat Vel	Heave	Pitch	Yaw	Roll
λ	-0.636± 0.305i	-1.063± 0.463i	-0.901	-2.900	-2.891	-3.668
u	1	0.096∓ 0.015i	-0.032	-.134^{x}	0.045	0.020
v	0.063± 0.039i	1	-0.029	-0.000	0.001	0.040^{x}
w	-0.083∓ 0.029i	0.071∓ 0.089i	1	0.021	-0.321	-0.015
p	0.057∓ 0.016i	1.668∓ 1.860i^{x}	0.011	-0.011	0.002	1
q	-0.568± 0.662i^{x}	-0.132± 0.139i	-0.008	1	-0.002	-0.136
r	-0.048∓ 0.115i	0.002∓ 0.038i	0.124	0.009	1	0.065
ϕ	-0.086∓ 0.005i	-1.962± 0.898i^{x}	-0.018	0.004	-0.017	-0.274^{R}
θ	1.134∓ 0.494i^{x}	0.152∓ 0.063i	0.002	-.345^{P}	-0.017	0.036

<u>Notes</u>: ^{x} denotes value free to float
^{P} denotes desired value of -0.345
^{R} denotes desired value of -0.286

V. Conclusions

The addition of cross coupling weighting to the algorithm developed by Captain Thomas Huckabone [2] allowed greater flexibility in achieving a desired closed loop eigenstructure for a linearized model of a combat helicopter at hovering conditions. The designer is able to use the algorithm to determine a full state regulator that yields desired performance, decoupling and acceptable robustness.

Non-dimensionalization of the input system was found to be necessary in achieving good results for the algorithm. The order of the desired eigenstructure was shown to alter the results of the algorithm. The convergence tolerance input to the program was shown to have an impact on the stability margins of the closed loop system. Also, the addition of cross coupling weights was shown to remove the guaranteed stability margins of the standard LQR solution.

VI. Recommendations

The algorithm provides several areas for further research. These include: program maintenance, expanding the application to non-hovering helicopters, validating the regulator solution via simulation and altering the algorithm to improve its robustness properties. Some of the following proposals should be incorporated into an extension of this and previous work as a Master's thesis at AFIT.

The source code in appendix C has not been properly reviewed with respect to operation and can probably be made much more efficient. A good start would be to eliminate the use of the DSVRGP subroutine from the IMSL package. In addition to improving efficiency, this will allow for the program to be used in a stand alone capacity on any computer system that uses the FORTRAN language. Additionally, the DLQGLIB package has exhibited some problems and should be altered or replaced.

This thesis has limited the application of the algorithm to two conventional combat helicopter models in hovering flight. In addition to looking at expanded flight conditions for helicopters, the algorithm should be applied to other multi-variable systems that can benefit from the application of optimal control design techniques.

Since full state feedback is not realistic in most cases, continued design efforts should be carried out to verify the performance and stability capabilities yielded by the algorithm via time response simulations. Additionally, the algorithm could be made to apply to observer design as well as regulator design.

As reported, the algorithm currently does not directly provide good stability margins when applying cross coupling weights. The addition of a stability parameter to the algorithm would correct this problem. Anderson and Moore [19:sect. 3.5] show that a prescribed degree of stability can be introduced to the standard regulator problem via a redefined LQR performance index

$$J = \int_0^\infty e^{2\sigma t} \left(x^T Q x + u^T R u \right) dt \qquad (66)$$

where σ specifies the minimum degree of stability of the resulting closed loop system. The algorithm could be altered to include σ as a designer chosen parameter, thus enforcing a robust algorithm solution.

Appendix A: AH-64 Model

Matrix A_{mo}:

```
Columns 1 through 3

  -2.8600000e-02   -6.3700000e-02    2.0500000e-02
   7.7900000e-02   -2.3100000e-01    5.9000000e-03
   4.6000000e-03   -2.5700000e-02   -2.6100000e-01
   5.6583750e-01   -3.5812500e+00    6.8043750e-01
   3.3663750e-01    8.4517500e-01    1.4325000e-02
   2.7933750e-01   -3.5096250e-01    5.7300000e-02
   0.0000000e+00    0.0000000e+00    0.0000000e+00
   0.0000000e+00    0.0000000e+00    0.0000000e+00

Columns 4 through 6

   3.19720768e-03    1.1127400e-01   -3.5881326e-03
  -1.15741710e-01   -1.4380454e-02   -2.2897033e-02
  -5.29144857e-03    3.1413613e-02    3.0575916e-02
  -2.70000000e+00   -1.3400000e-01   -6.6200000e-01
  -9.20000000e-03   -7.5000000e-01    2.4400000e-02
  -1.05000000e+00    4.1300000e-01   -4.0000000e-01
   1.00000000e+00   -5.1000000e-03    1.0300000e-01
   0.00000000e+00    9.9900000e-01    4.9900000e-02

Columns 7 through 8

   0.0000000e+00         -4.4677138e-01
   4.4677138e-01          2.2897033e-03
   2.2338569e-02         -4.5794066e-02
   0.0000000e+00          0.0000000e+00
   0.0000000e+00          0.0000000e+00
   0.0000000e+00          0.0000000e+00
   0.0000000e+00          0.0000000e+00
   0.0000000e+00          0.0000000e+00
```

<u>Matrix B_{me}</u>:

```
  Columns 1 through 3

     1.5660000e-01      3.4560000e-01     -3.9900000e-02
    -5.6936937e-02      8.1194030e-02      1.7180000e-01
    -1.5372000e-01      3.4500000e-02     -8.7000000e-03
    -1.1294000e+00     -2.5785000e+00      1.6219500e+01
     1.8570000e-01     -4.3405000e+00     -2.2562000e+00
     2.0628000e+00      4.1690000e-01      5.0137000e+00
     0.0000000e+00      0.0000000e+00      0.0000000e+00
     0.0000000e+00      0.0000000e+00      0.0000000e+00

  Column 4

    -6.6666667e-04
     2.0870000e-01
     8.8888889e-04
     4.2402000e+00
     1.0070000e-01
     2.4116000e+00
     0.0000000e+00
     0.0000000e+00
```

Appendix B: UH-60 Mathematical Model

Matrix A:

```
Columns 1 through 4

-2.1100e-02  -1.3200e-02   1.2300e-02  -1.2081e+00
 4.8000e-03  -2.0700e-02            0  -2.5520e-01
-5.0000e-04  -5.0000e-04  -2.3560e-01  -9.7700e-02
 3.6800e-02  -3.1200e-02   2.3000e-03  -5.7728e+00
 3.8000e-03   6.7000e-03   1.1000e-03   1.5320e-01
 6.0000e-04   4.1000e-03            0  -6.3800e-02
          0            0            0   1.0000e+00
          0            0            0            0

Columns 5 through 8

 2.0151e+00  -2.8160e-01  -3.6500e-02  -3.2066e+01
-8.6590e-01  -1.3799e+00   3.2028e+01   7.5000e-02
 1.9317e+00   2.2755e+00   1.6168e+00  -1.4828e+00
-1.6579e+00   1.4310e-01   4.5500e-02            0
-9.0940e-01  -1.8500e-02   1.3000e-02            0
-1.1250e-01  -2.2380e-01   1.0000e-03            0
-2.3000e-03   4.6200e-02            0            0
 9.9870e-01   5.0500e-02            0            0
```

Matrix B:

```
 5.8590e-01  -1.4993e+00  -7.1600e-02   9.5370e-01
 1.0980e-01  -3.7400e-02   4.1700e-01  -8.7780e-01
-6.4483e+00  -6.5900e-02  -2.2800e-02   4.3320e-01
-2.1450e-01  -9.8000e-03   1.3989e+00  -5.9880e-01
-3.7700e-02   3.4830e-01   2.2800e-02  -9.4000e-03
 1.0900e-01   1.3200e-02   2.3300e-02   3.9980e-01
          0            0            0            0
          0            0            0            0
```

```
      PROGRAM EIGSPACE
      IMPLICIT COMPLEX*16 C
      IMPLICIT REAL*8 (A-B,D-H,O-Z)
      COMMON /INOU/KIN,KOUT
      COMMON A,B,ed,ea,G,NR,NA,ND,M,N,NN,ACL,P,EV,
     + WDES,WACH,calpha,iwrite,nrcode,nscode
      DIMENSION X(80),A(10,10),B(10,10),R(10,10),Q(10,10),
     1 RK(10,10),G(10,10),ACL(10,10),P(10,10),EV(10)
      DIMENSION XGUESS(80),XS(80),GRAD(80),X2(80)
      DIMENSION v(80,81),Fvec(81),vs(80),vss(80,81),
     + Fvec1(81)
      DIMENSION EVS(10),PS(10,10),S(10,10)
      DIMENSION edr(10),edi(10),wdesr(10,10),wdesi(10,10)
      REAL*8 maged(10)
      COMPLEX*16 ea(10),edg(10),WDES(10,10),WACH(10,10)
      COMPLEX*16 ed(10),WDESS(10,10),calpha(10)
      INTEGER IPERMM(81),IPERMD(10)
      EXTERNAL PPFUNC,DMACH,DUMCGF
      open(UNIT=10,FILE='input.dat',STATUS='old')
      open(UNIT=9,FILE='output.dat',STATUS='old')
      rewind 10
      rewind 9
      KIN=5
      KOUT=6
c-----
c     read model size and set integers for array sizes
c-----
      iwrite=0
      read(10,*) N
      NA=N
      NN=2*N
      NA2=N*N
      ND=NN*(4*N+3)
c-----
c     read values for A and B matrices
c-----
      jj=1
      icount=0
      do 10 i=1,NA2
        icount=icount+1
        if(icount.eq.11) then
          jj=jj+1
          icount=1
        endif
      read(10,*) A(icount,jj)
   10 continue
      read(10,*) M
      NR=M
      NB=N*M
      jj=1
      icount=0
      do 20 i=1,NB
        icount=icount+1
        if(icount.eq.11) then
          icount=1
          jj=jj+1
        endif
        read(10,*) B(icount,jj)
   20 continue
c-----
c     read the desired eigenstructure and weights
c-----
      do 30 i=1,N
        read(10,*) EV(i),edr(i),edi(i)
        ed(i)=DCMPLX(edr(i),edi(i))
   30 continue
      jj=1
      icount=0
      do 35 i=1,NA2
        icount=icount+1
        if(icount.eq.11) then
          icount=1
          jj=jj+1
        endif
      read(10,*) P(icount,jj),wdesr(icount,jj),
     + wdesi(icount,jj)
      WDES(icount,jj)=DCMPLX(wdesr(icount,jj),
     + wdesi(icount,jj))
   35 continue
      call WNORM(WDES,N)
c-----
c     read tolerances and codes
c-----
      read(10,*) tol
      ievalmax=1000
      read(10,*) nrcode
      read(10,*) nscode
      read(10,*) kmax
c-----
c     set initial guess for R and Q.  Use the identity
c     matrix in both cases.  Put the upper triangular
c     portion of each in XGUESS
c-----
      ix=0
      if(nrcode.eq.1) then
        XGUESS(1)=1.0d0
        ix=1
        goto 51
      endif
      if(nrcode.eq.2) then
        do 41 i=1,M
          ix=ix+i
          XGUESS(ix)=1.0d0
   41   continue
      else
        icount=0
        do 50 i=1,M
          icount=icount+1
          do 40 jj=icount,M
            ix=ix+1
            if(icount.eq.jj)then
              XGUESS(ix)=1.0d0
            else
              XGUESS(ix)=0.0d0
            endif
   40     continue
   50   continue
      endif
   51 continue
      icount=0
      do 70 i=1,N
        icount=icount+1
        do 60 jj=icount,N
          ix=ix+1
          if(icount.eq.jj)then
            XGUESS(ix)=1.0d0
          else
            XGUESS(ix)=0.0d0
          endif
   60   continue
   70 continue
c-----
c     add S matrix parameters to end of XGUESS if applicable
c-----
      if(nscode.ne.0) then
        do 80 i=1,M
          do 85 jj=1,N
            ix=ix+1
            XGUESS(ix)=0.0d0
   85     continue
   80   continue
      endif
c-----
c     initialize X,Fvec,Fvec1,vs and vss
c-----
      do 91 i=1,ix
        X(i)=0.0d0
        vs(i)=0.0d0
   91 continue
      ixp1=ix+1
      do 93 i=1,ixp1
        Fvec(i)=0.0d0
        Fvec1(i)=0.0d0
        do 92 jj=1,ix
          vss(jj,i)=0.0d0
   92   continue
   93 continue
c-----
c     first iteration, change XGUESS to first solution (X)
c-----
      CALL FMINS(ix,XGUESS,X,tol,ixp1,v,Fvec1,vs,vss,
     + IPERMM,ievalmax)
      kcount=0
      do 100 i=1,ix
        XGUESS(i)=X(i)
  100 continue
c-----
c     begin subsequent iterations
c-----
  110 continue
      kcount=kcount+1
      do 121 i=1,ix
        X2(i)=0.0d0
        vs(i)=0.0d0
  121 continue
      do 123 i=1,ixp1
        Fvec(i)=0.0d0
        do 122 jj=1,ix
          vss(jj,i)=0.0d0
  122   continue
  123 continue
      call FMINS(ix,XGUESS,X2,tol,ixp1,v,Fvec,vs,
     + vss,IPERMM,ievalmax)
      do 130 i=1,ix
        XGUESS(i)=X2(i)
  130 continue
c-----
c     check against tolerance
c-----
      WRITE (*,*) kcount,Fvec1(1),Fvec(1)
      delJ=dabs(Fvec1(1)-Fvec(1))
      Fvec1(1)=Fvec(1)
      if (delJ.gt.tol.and.kcount.lt.50) goto 110
c-----
c     final iteration using last XGUESS
c-----
      iwrite=1
      CALL PPFUNC(ix,XGUESS,RJ)
      do 238 i=1,N
        write (*,*) ea(i)
  238 continue
      end
      SUBROUTINE FMINS(NX,XGUESS,X,TOL,NXP1,v,Fvec,vs,
     + vss,IPERMM,ievalmax)
      IMPLICIT COMPLEX*16 C
      IMPLICIT REAL*8 (A-B,D-H,O-Z)
      DIMENSION XGUESS(NX),X(NX),v(NX,NXP1),Fvec(NXP1),
     + vs(NX),vss(NX,NXP1),vr(80),vk(80),ve(80),
     + vt(80),vc(80),vbar(80)
      INTEGER IPERMM(NXP1)
      i=100
      icallf=0
      icount=0
c-----
c     Build initial simplex near XGUESS
c          v(i,j)=simplex matrix
c          vs(i)=scratch vector
c          Fvec(i)=function values corresponding to v(i,j)
c                  columns
c-----
      xnx=dflotj(NX)
      aa=0.5d0
      p=aa*(dsqrt(xnx+1.0d0)+xnx-1.0d0)/(xnx*dsqrt(2.0d0))
      q=aa*(dsqrt(xnx+1.0d0)-1.0d0)/(xnx*dsqrt(2.0d0))
      do 1010 i=1,NX
        v(i,1)=XGUESS(i)
        vs(i)=v(i,1)
        X(i)=XGUESS(i)
 1010 continue
      icallf=icallf+1
      call ppfunc(NX,vs,Fv)
      Fvec(1)=Fv
      i=1
      do 1040 jj=1,NX
        do 1020 kk=1,NX
          vs(kk)=X(kk)
 1020   continue
        i=jj+1
        do 1030 kk=1,NX
          if(jj.eq.kk) then
            v(kk,i)=vs(kk)+p
          else
            v(kk,i)=vs(kk)+q
          endif
          vs(kk)=v(kk,i)
 1030   continue
        icallf=icallf+1
        call PPFUNC(NX,vs,Fv)
        Fvec(i)=Fv
 1040 continue
c-----  sort the simplex in ascending order
c          IPERMM(i) = vector of index of sorted simplex
c                      sort is in ascending order
c-----        vsum(i) = summation of abs(v(:,i)
      do 1050 i=1,NXP1
```

```
         IPERMM(i)=i
 1050 continue
      call DSVRGP(NXP1,Fvec,Fvec,IPERMM)
      do 1070 i=1,NXP1
        do 1060 jj=1,NX
          vss(jj,i)=v(jj,i)
 1060   continue
 1070 continue
      do 1090 i=1,NXP1
        do 1080 jj=1,NX
          v(jj,i)=vss(jj,ipermm(i))
 1080   continue
 1090 continue
 1100 continue
      if(icount.gt.ievalmax) goto 1130
      test=0.0d0
      vsum=0.0d0
      do 1120 i=2,NXP1
        do 1110 jj=1,NX
          vsum=dabs(v(jj,i)-v(jj,1))+vsum
 1110   continue
        test=dmax1(test,vsum)
 1120 continue
      if(test.le.tol) go to 1130
c-----
c       initialize vr,vk,ve,vt,vs,vss,vc,vbar
c-----
      do 1121 i=1,NX
        vr(i)=0.0d0
        vk(i)=0.0d0
        ve(i)=0.0d0
        vt(i)=0.0d0
        vs(i)=0.0d0
        vc(i)=0.0d0
        vbar(i)=0.0d0
        do 1122 jj=1,NXP1
          vss(i,jj)=0.0d0
 1122   continue
 1121 continue
      call FMINSTEP(v,NX,NXP1,Fvec,vr,vk,ve,vt,vs,vss,
     + vc,vbar,IPERMM)
      icount=icount+1
      goto 1100
 1130 continue
      do 1140 i=1,NX
        X(i)=v(i,1)
 1140 continue
      return
      end
      SUBROUTINE FMINSTEP(v,NX,NXP1,Fvec,vr,vk,ve,
     + vt,vs,vss,vc,vbar,IPERMM)
      IMPLICIT COMPLEX*16 C
      IMPLICIT REAL*8 (A-B,D-H,O-Z)
      DIMENSION v(NX,NXP1),Fvec(NXP1),vr(NX),vk(NX),
     + ve(NX),vt(NX),vs(NX),vss(NX,NXP1),vc(NX),
     + vbar(NX)
      INTEGER IPERMM(NXP1)
      icall=0
      alpha=1.0d0
      beta=0.5d0
      gamma=2.0d0
      xnx=dflotj(NX)
      do 2020 i=1,NX
        vb=0.0d0
        do 2010 jj=1,NX
          vb=vb+v(i,jj)
 2010   continue
        vbar(i)=vb/xnx
 2020 continue
      do 2030 i=1,NX
        vr(i)=vbar(i)+alpha*(vbar(i)-v(i,NXP1))
 2030 continue
      icall=icall+1
      call PPFUNC(NX,vr,fr)
      do 2040 i=1,NX
        vk(i)=vr(i)
 2040 continue
      fk=fr
      if(fr.lt.Fvec(1)) then
          do 2050 i=1,NX
            ve(i)=vbar(i)+gamma*(vr(i)-vbar(i))
 2050     continue
          icall=icall + 1
          call PPFUNC(NX,ve,fe)
          if(fe.lt.Fvec(1)) then
            do 2060 i=1,NX
              vk(i)=ve(i)
 2060       continue
            fk=fe
          endif
      else
        if(fr.ge.Fvec(NXP1)) then
          do 2070 i=1,NX
            vt(i)=v(i,NXP1)
 2070     continue
          ft=Fvec(NXP1)
        else
          do 2080 i=1,NX
            vt(i)=vr(i)
 2080     continue
          ft=fr
        endif
```

```
         do 2090 i=1,NX
           vc(i)=vbar(i)-beta*(vbar(i)-vt(i))
 2090    continue
         icall=icall+1
         call PPFUNC(NX,vc,fc)
         if(fc.lt.Fvec(NX)) then
           if(fc.ge.fr) goto 2135
           do 2100 i=1,NX
             vk(i)=vc(i)
 2100      continue
           fk=fc
         else
           do 2120 i=2,NX
             do 2110 jj=1,NX
               v(jj,i)=(v(jj,1)+v(jj,i))/2.0d0
               vs(jj)=v(jj,i)
 2110        continue
             icall=icall+1
             call PPFUNC(NX,vs,Fv)
             Fvec(i)=Fv
 2120      continue
           do 2130 i=1,NX
             vk(i)=(v(i,1)+v(i,NXP1))/2.0d0
 2130      continue
           icall=icall+1
           call PPFUNC(NX,vk,fk)
 2135    endif
       endif
       do 2140 i=1,NX
         v(i,NXP1)=vk(i)
 2140  continue
       Fvec(NXP1)=fk
       do 2150 iii=1,NXP1
         IPERMM(iii)=iii
 2150  continue
       call DSVRGP(NXP1,Fvec,Fvec,IPERMM)
       do 2170 i=1,NXP1
         do 2160 jj=1,NX
           vss(jj,i)=v(jj,i)
 2160    continue
 2170  continue
       do 2190 i=1,NXP1
         do 2180 jj=1,NX
           v(jj,i)=vss(jj,IPERMM(i))
 2180    continue
 2190  continue
       return
       end
```

```
      SUBROUTINE PPFUNC(NX,X,RJ)
      IMPLICIT COMPLEX*16 C
      IMPLICIT REAL*8 (A-B,D-H,O-Z)
      COMMON /INOU/KIN,KOUT
      COMMON A,B,ed,ea,G,NR,NA,ND,M,N,NN,ACL,P,EV,
     + WDES,WACH,calpha,iwrite,nrcode,nscode
      DIMENSION X(80),A(10,10),B(10,10),R(10,10),Q(10,10),
     1 RK(10,10),G(10,10),ACL(10,10),P(10,10),EV(10),evs(10)
      DIMENSION RCOPY(10,10),IPVT(10),WORK(10)
      DIMENSION RIST(10,10),BRIST(10,10),SRIST(10,10)
      DIMENSION GK(10,10)
      DIMENSION ANEW(10,10),QNEW(10,10),BK(10,10)
      DIMENSION DUM(860,1),IDUM(20),WR(10),WI(10),Z(10,10),
     1 IV1(10),FV1(10),ACLS(10,10),S(10,10)
      COMPLEX*16 ea(10),WDES(10,10),WACH(10,10),
     1 WDESS(10,10),WACHS(10,10),calpha(10)
      COMPLEX*16 ed(10),cedif(10),edtmp(10),eatmp(10)
      REAL*8 magea(10)
c-----
c     this subroutine calls the cost function subroutine,
c     and allows variable arrays to be set
c-----
      RJ=0.0d0
      do 176 i=1,10
        ea(i)=DCMPLX(0.0d0,0.0d0)
        do 177 jj=1,10
          ACL(i,jj)=0.0d0
          WACH(i,jj)=dcmplx(0.0d0,0.0d0)
          G(i,jj)=0.0d0
  177 continue
  176 continue
      call PP(NX,X,RJ,NR,NA,ND,N,M,NN,A,B,R,Q,S,RK,G,ACL,P,
     1 EV,evs,DUM,IDUM,WR,WI,Z,IV1,FV1,ea,ed,WDES,WACH,
     2 cedif,edtmp,eatmp,magea,ACLS,WDESS,WACHS,calpha,
     3 iwrite,nrcode,nscode,RCOPY,IPVT,WORK,RIST,BRIST,
     4 SRIST,GK,ANEW,QNEW,BK)
      RETURN
      END
      SUBROUTINE PP(NX,X,RJ,NR,NA,ND,N,M,NN,A,B,R,Q,S,RK,G,
     1 ACL,P,EV,evs,DUM,IDUM,WR,WI,Z,IV1,FV1,ea,ed,WDES,
     2 WACH,cedif,edtmp,eatmp,magea,ACLS,WDESS,WACHS,calpha,
     3 iwrite,nrcode,nscode,RCOPY,IPVT,WORK,RIST,BRIST,
     4 SRIST,GK,ANEW,QNEW,BK)
      IMPLICIT COMPLEX*16 C
      IMPLICIT REAL*8 (A-B,D-H,O-Z)
      COMMON /INOU/KIN,KOUT
      DIMENSION X(NX),A(N,N),B(N,M),R(M,M),Q(N,N),
     1 RK(N,N),G(M,N),ACL(N,N),P(N,N),EV(N),WNORMA(10),
     2 PS(10,10),EVS(N),S(N,M)
      DIMENSION DUM(ND,1),IDUM(NN),WR(N),WI(N),Z(N,N),
     1 IV1(N),FV1(N)
      DIMENSION RM(10,10),QH(10,10),SN(10,10),SNT(10,10)
      DIMENSION S1(10,10),S2(10,10),QS(10,10),SR(10,10)
      DIMENSION er(10),ei(10)
      DIMENSION edifmag(10),ACLS(N,N)
      DIMENSION PDUM(10,10),EVDUM(10,10)
      DIMENSION RCOPY(M,M),IPVT(M),WORK(M)
      DIMENSION RIST(M,N),BRIST(N,N),SRIST(N,N)
      DIMENSION GK(M,N)
      DIMENSION ANEW(N,N),QNEW(N,N),BK(M,N)
      DIMENSION QSAVE(10,10),RSAVE(10,10)
      COMPLEX*16 ea(N),ed(N),WDES(N,N),WACH(N,N),WDESS(N,N),
     1 WACHS(N,N),calpha(N)
      COMPLEX*16 cedif(N),edtmp(N),eatmp(N)
      INTEGER IPERMA(10),imin(10)
      REAL*8 magea(10)
      LOGICAL ELIM
      IPRT=0
c-----
c     if last iteration, write to output
c-----
      if(iwrite.ne.0) then
        write (9,*) N
        write (9,*) M
      endif
c-----
c     set the Q, R and S matrices
c-----
      CALL MAKEQRS(N,M,NX,X,Q,R,S,RM,QH,SN,SNT,S1,S2,QS,SR,
     1 nrcode,nscode,QSAVE,RSAVE)
      if(iwrite.ne.0) then
        do 557 i=1,M
          do 556 jj=1,M
            write (9,*) R(jj,i)
  556     continue
  557   continue
        do 555 i=1,N
          do 554 jj=1,N
            write (9,*) Q(jj,i)
  554     continue
  555   continue
```

```
        do 553 i=1,M
          do 552 jj=1,N
            write (9,*) S(jj,i)
  552     continue
  553   continue
      endif
c-----
c     calculate the lqr gain matrix, GK
c-----
      CALL TRNATB(N,M,N,M,S,RIST)
      CALL SAVE(M,M,M,M,R,RCOPY)
      CALL MLINEQ(M,M,N,RCOPY,RIST,NCOND,IPVT,WORK,1)
      CALL MMUL(N,M,N,N,M,N,B,RIST,BRIST)
      CALL MMUL(N,M,N,N,M,N,S,RIST,SRIST)
      CALL MSUB(N,N,N,N,N,A,BRIST,ANEW)
      CALL MSUB(N,N,N,N,N,Q,SRIST,QNEW)
      CALL REG(NA,NR,N,M,NN,ANEW,B,QNEW,R,RK,G,ACL,DUM,IDUM,
     1 IPRT)
      CALL MADD(M,M,M,M,N,G,RIST,GK)
      if(iwrite.ne.0) then
        do 530 i=1,N
          do 5555 jj=1,M
            write (9,*) GK(jj,i)
 5555     continue
  530   continue
      endif
c-----
c     calculate the new closed loop eigenvalues
c-----
      CALL MMUL(N,M,N,N,M,N,B,GK,BK)
      CALL MSUB(N,N,N,N,N,A,BK,ACLS)
      ipc=1
      CALL EIGVV(NA,N,ACLS,WR,WI,Z,IV1,FV1,IPC,IERR)
c-----
c     count complex pairs and configure eigenvectors
c-----
      icomplex=0
      do 539 i=1,N
        if(WI(i).ne.0.0d0) icomplex=icomplex+1
  539 continue
      NCMP=N-icomplex/2
      ii=0
      do 540 i=1,NCMP
        ii=ii+1
          if(dabs(WI(ii)).gt.0.0) then
            do 535 jj=1,N
              WACH(jj,ii)=DCMPLX(Z(jj,ii),Z(jj,ii+1))
              WACH(jj,ii+1)=DCONJG(WACH(jj,ii))
  535       continue
            ii=ii+1
          else
            do 536 jj=1,N
              WACH(jj,ii)=DCMPLX(Z(jj,ii),0.0d0)
  536       continue
          endif
  540   continue
        call WNORM(WACH,N)
        do 30 i=1,N
         ea(i)=dcmplx(WR(i),WI(i))
   30   continue
c-----
c       write to output if final iteration
c-----
   40   continue
        if(iwrite.ne.0) then
          do 43 i=1,N
            eareal=dreal(ea(i))
            eaimag=dimag(ea(i))
            write (9,*) eareal
            write (9,*) eaimag
   43     continue
        endif
        do 41 i=1,N
          eatmp(i)=ea(i)
          edtmp(i)=ed(i)
          EVS(i)=EV(i)
          do 42 jj=1,N
            WACHS(jj,i)=WACH(jj,i)
            WDESS(jj,i)=WDES(jj,i)
            PS(jj,i)=P(jj,i)
            if(iwrite.ne.0) then
              wreal=dreal(wach(jj,i))
              wimag=dimag(wach(jj,i))
              write (9,*) wreal
              write (9,*) wimag
            endif
   42   continue
   41   continue
        jjj=0
        do 501 jj=1,N
          do 502 kk=1,N
            if(P(jj,kk).ne.0.0d0) jjj=jjj+1
  502     continue
  501   continue
        NL=N
        RJ=0.0d0
        do 50 i=1,N
c-----
c       calculate difference between desired eigenvalues and
c       achievable eigenvalues.  algorithm matches the closest
c       achieved eigenvalue to the desired eigenvalues, in the
c       order they were input.
c-----
          do 51 jj=1,NL
            cedif(jj)=edtmp(1)-eatmp(jj)
   51     continue
```

```
        do 56 jj=1,NL
          edifmag(jj)=dsqrt(dreal(cedif(jj))**2+
     +                    dimag(cedif(jj))**2)
   56   continue
        call imins(NL,edifmag,imin(i))
        iii=IMIN(i)
        RJVEC=0.0d0
        if(jjj.eq.0) then
          if(iwrite.ne.0) then
            goto 503
          endif
        endif
c-----
c     calculate the calpha for the eigenvectors
c-----
        sum1=0.0d0
        sum2=0.0d0
        if(dimag(eatmp(iii)).ne.0.0d0) then
          do 584 jj=1,N
            sum1 = sum1+dimag(WDESS(jj,1))
     +             *dreal(WACHS(jj,iii))
     +             -dreal(WDESS(jj,1))*dimag(WACHS(jj,iii))
            sum2 = sum2+dreal(WDESS(jj,1))
     +             *dreal(WACHS(jj,iii))
     +             +dimag(WDESS(jj,1))*dimag(WACHS(jj,iii))
  584     continue
          ph1=datan2(sum1,sum2)
          calpha1=dcmplx(dcos(ph1),dsin(ph1))
          calpha2=-1*calpha1
        else
          calpha1=(1.0d0,0.0d0)
          calpha2=-1*calpha1
        endif
c-----
c     determine which calpha produces minimum cost and
c     calculate eigenvector contribution to performance
c     index
c-----
        DELWI1=0.0d0
        DELWI2=0.0d0
          do 585 jj=1,N
            DELWI1=PS(jj,1)*((DREAL(WDESS(jj,1)-
     +           calpha1*WACHS(jj,iii)))**2
     +          +(DIMAG(WDESS(jj,1)-calpha1
     +          *WACHS(jj,iii)))**2)
     +          +DELWI1
            DELWI2=PS(jj,1)*((DREAL(WDESS(jj,1)-
     +           calpha2*WACHS(jj,iii)))**2
     +          +(DIMAG(WDESS(jj,1)-calpha2
     +          *WACHS(jj,iii)))**2)
     +          +DELWI2
  585     continue
          if(DELWI1.lt.DELWI2) then
```

```
            DELWI=DELWI1
            calphai=calpha1
          else
            DELWI=DELWI2
            calphai=calpha2
          endif
        RJVEC=DELWI
        if (iwrite.ne.0) then
          do 587 jj=1,N
            if(eatmp(iii).eq.ea(jj)) calpha(jj)=calphai
  587     continue
          if(i.eq.N)then
            do 586 jj=1,N
              alreal=dreal(calpha(jj))
              alimag=dimag(calpha(jj))
  586       continue
          endif
        endif
  503   continue
c-----
c     add eigenvalue/eigenvector contribution to performance
c     index
c-----
         RJ=RJ+EVS(1)*edifmag(iii)**2+RJVEC
c-----
c     reset the eigenvalue/vector arrays to eliminate
c     those poles already matched.
c-----
        k=0
        NL=NL-1
        do 52 jj=1,NL
          k=k+1
          ELIM=jj.eq.iii
          IF(ELIM) K=k+1
          eatmp(jj)=eatmp(k)
          edtmp(jj)=edtmp(jj+1)
          EVS(jj)=EVS(jj+1)
          if(jjj.eq.0) goto 591
          do 590 kk=1,N
            WACHS(kk,jj)=WACHS(kk,k)
            WDESS(kk,jj)=WDESS(kk,jj+1)
            PS(kk,jj)=PS(kk,jj+1)
  590     continue
  591     continue
   52   continue
   50 continue
      RETURN
      END
      SUBROUTINE IMINS(NL,EDIFMAG,I)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION EDIFMAG(NL)
      i=1
      do 5000 jj=1,nl
        if(dabs(edifmag(jj)).lt.dabs(edifmag(i))) i=jj
 5000 continue
      return
      end
      SUBROUTINE WNORM(WVEC,N)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION WNORMV(10)
      COMPLEX*16 WVEC(N,N)
      do 5 i=1,N
       WNORMV(i)=0.0d0
    5 continue
      do 10 i=1,N
       do 20 jj=1,N
        WNORMV(i)=WNORMV(i)+(dreal(WVEC(jj,i)))**2+
     +            (dimag(WVEC(jj,i)))**2
   20  continue
   10 continue
      do 30 i=1,N
       do 40 jj=1,N
        WVEC(jj,i)=WVEC(jj,i)/dsqrt(wnormv(i))
   40  continue
   30 continue
      return
      end
      SUBROUTINE MAKEQRS(N,M,NX,X,Q,R,S,RM,QH,SN,SNT,S1,S2,
     1 QS,SR,nrcode,nscode,QSAVE,RSAVE)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION X(NX),Q(N,N),R(M,M),S(N,M)
      DIMENSION RM(M,M),QH(N,N),SN(N,M),SNT(M,N)
      DIMENSION S1(N,N),S2(M,M),QS(N,M),SR(N,M),QSAVE(N,N),
      DIMENSION RSAVE(M,M)
c-----
c     this subroutine reads the upper triangular portion
c     of matrices RM, QH and all of SN, if applicable, from
c     X and returns Q, R and S matrices that meet the
c     constraints necessary for an LQR solution
c-----
      ix=0
      if(nrcode.eq.1.or.nrcode.eq.2) then
        ix=1
        do 116 i=1,M
          do 117 jj=1,M
            if(i.eq.jj) then
              RM(i,jj)=X(ix)
            else
              RM(i,jj)=0.0d0
            endif
 117      continue
          if(nrcode.eq.2) ix=ix+1
 116    continue
      else
        icount=0
        do 101 i=1,M
          icount=icount+1
          do 102 jj=icount,M
            ix=ix+1
            RM(i,jj)=X(ix)
            RM(jj,i)=X(ix)
 102      continue
 101    continue
      endif
      icount=0
      do 111 i=1,N
        icount=icount+1
        do 112 jj=icount,N
        ix=ix+1
          QH(i,jj)=X(ix)
          QH(jj,i)=X(ix)
 112  continue
 111  continue
      if(nscode.ne.0) then
        do 131 i=1,M
          do 132 jj=1,N
            ix=ix+1
            SN(jj,i)=X(ix)
            SNT(i,jj)=X(ix)
 132      continue
```

```
131   continue
    endif
      call MMUL(N,N,N,N,N,N,QH,QH,Q)
      call MMUL(M,M,M,M,M,M,RM,RM,R)
    if(nscode.eq.0) then
      call ZPART(N,N,M,S)
    else
      call MMUL(N,M,N,N,M,N,SN,SNT,S1)
      call MMUL(M,N,M,M,N,M,SNT,SN,S2)
      call MMUL(N,N,N,N,N,M,QH,SN,QS)
      call MMUL(N,M,M,N,M,M,SN,RM,SR)
      call SAVE(N,N,N,N,Q,QSAVE)
      call MADD(N,N,N,N,N,QSAVE,S1,Q)
      call SAVE(M,M,M,M,R,RSAVE)
      call MADD(M,M,M,M,M,S2,RSAVE,R)
      call MADD(N,N,N,N,M,QS,SR,S)
    endif
    return
    end
```

Appendix D: Operating Instructions

General

The following is provided as a guide for using the included algorithm at AFIT. While the source code is written in a standard FORTRAN language [24], the program requires some special handling. Particularly, the specific subroutines called upon in the program must be properly accessed. The procedures presented here ensure proper use of the program. The program is run on the Virtual Memory System (VMS) cluster at AFIT. The cluster includes three host computers: Hercules, Lancer and Sabre. Reference [25] contains more details about the AFIT computer services.

Compiling

The EIGSPACE source code is compiled using the FORTRAN compiler on VMS. The object file is linked to the DLQGLIB and IMSL subroutine packages. The DLQGLIB source code is available at AFIT. For this project, the DLQGLIB object file was placed in the same directory as the EIGSPACE program. The IMSL subroutines must also be linked to the algorithm. These routines are not available in FORTRAN at AFIT. Following are the commands used to compile the algorithm:

```
FORTRAN eigspace
LINK eigspace,dlqglib,imslib_share/opt
```

MATLAB™ Interface

The EIGSPACE program requires a single input file and a single output file. Since most of the input and output are expressed as matrices, MATLAB™ provides a convenient interface for the program. MATLAB™ can be used as a stand alone interface, via the LQREA.M file, which requires interactive use of a terminal. The program can be more efficiently run when submitted via the following command file EIGSPACE.COM:

```
$run EIGSPACE
$exit
```

The file, EIGSPACE.LOG is created which contains all screen output data from the algorithm as well as other information about the program execution. The LQRSAVE.M file is used to create the input file for EIGSPACE, while the LQROUT.M file is used to reformat the output file. The input and output files should be in the same directory as the EIGSPACE file. The m-files presented below ensure that the program will run properly when MATLAB™ is used in the same directory.

LQREA m-file

```
function[Q,R,S,ea,va,K]=LQREA(a,b,ed,Fe,vd,Fv,tol,rcode,scode,
                                    kmax)
%LQREA Eigenstructure assignment using the Linear Quadratic
%Regulator.
%
%Form is [Q,R,S,ea,va,K]=
%       LQREA(a,b,ed,Fe,vd,Fv,tol,rcode,scode,kmax)
%
%Input parameters,
%  a,b=state space matrices of a linear system with n states
%      and m controls
%  ed=nxn diagonal matrix of desired eigenvalues
%  Fe=nx1 matrix weighting each eigenvalue
%  vd=nxn matrix whose columns are the desired eigenvectors
%     (must be in same order as associated eigenvalues)
%  Fv=nxn matrix whose elements weight corresponding elements
%     of vd
%  tol=convergence tolerance for performance index
%  rcode=(1) R=ro*I, (2) R=[diag], (other) R=positive definite
%  scode=(0) S=nxm zero matrix, (other) S=control-state
%                                          weights
%  kmax=maximum optimization iterations
%Output parameters,
%  ea=nxn diagonal matrix of achievable eigenvalues
%  va=nxn matrix of achievable eigenvectors
%  Q,R,S=final state and control weighting matrices
%  K=gain matrix
%
[n,nc]=size(a);
[mr,m]=size(b);
at=a(:);
bt=b(:);
for i=1:n,
  edd(i,1)=Fe(i);
  edd(i,2)=real(ed(i,i));
  edd(i,3)=imag(ed(i,i));
end
vdd(:,1)=Fv(:);
vdd(:,2)=real(vd(:));
vdd(:,3)=imag(vd(:));
save input.dat n at m bt edd vdd tol rcode scode kmax /ascii
!run eigspace
load output.dat
count=1;
n=output(count);
count=count+1;
m=output(count);
count=count+1;
n2=n*n;
m2=m*m;
nm=n*m;
```

```
R=zeros(m,m);
R(:)=output(count:count+m2-1);
count=count+m2;
Q=zeros(n,n);
Q(:)=output(count:count+n2-1);
count=count+n2;
S=ones(n,m);
S(:)=output(count:count+nm-1);
count=count+nm;
K=zeros(m,n)
K(:)=output(count:count+nm-1)
count=count+nm
ea=zeros(n,n);
for i=1:n
 ea(i,i)=output(count)+j*output(count+1);
 count=count+2;
end
va=zeros(n,n);
for jj=1:n
 for i=1:n
  va(i,jj)=output(count)+j*output(count+1);
  count=count+2;
 end
end
```

LQRSAVE m-file

```
function LQRSAVE(a,b,ed,Fe,vd,Fv,tol,rcode,scode,kmax)
%LQRSAVE saves all required data for the Eigenstructure
%assignment algorithm using the Linear Quadratic Regulator.
%
%     Form is LQRSAVE(a,b,ed,Fe,vd,Fv,tol,rcode,scode)
%     Input parameters,
%           a,b = state space matrices of a linear system with
%                 n states and m controls
%           ed = nxn diagonal matrix of desired eigenvalues
%           Fe = nx1 matrix weighting each eigenvalue
%           vd = nxn matrix whose columns are the desired
%                eigenvectors (must be in same order as
%                associated eigenvalues)
%           Fv = nxn matrix whose elements weight
%                corresponding elements of vd
%           tol = convergence tolerance for performance index
%           rcode = (1) R = ρI
%                   (2) R = [diag]
%                   (3) R = positive definite
%           scode = (0) S=[0]
%                   (1) S=control-state weights
%           kmax = maximum optimization iterations
[n,nc]=size(a);
[mr,m]=size(b);
at=a(:);
bt=b(:);
for i=1:n,
  edd(i,1)=Fe(i);
  edd(i,2)=real(ed(i,i));
  edd(i,3)=imag(ed(i,i));
end
vdd(:,1)=Fv(:);
vdd(:,2)=real(vd(:));
vdd(:,3)=imag(vd(:));
save input.dat n at m bt edd vdd tol rcode scode kmax /ascii
```

LQROUT m-file

```
function [Q,R,S,ea,va,K]=LQROUT
%LQROUT loads output from EIGNEW eigenstructure assignment
% algorithm  using the Linear Quadratic Regulator.
%
%    Form is [Q,R,S,ea,va,K]=LQROUT
%
%       Output parameters,
%         Q,R,S=final state and control weighting matrices
%         ea=nxn diagonal matrix of achievable eigenvalues
%         va=nxn matrix of achievable eigenvectors
%         K=gain matrix
%
load output.dat
count=1;
n=output(count);
count=count+1;
m=output(count);
count=count+1;
n2=n*n;
m2=m*m;
nm=n*m;
R=zeros(m,m);
R(:)=output(count:count+m2-1);
count=count+m2;
Q=zeros(n,n);
Q(:)=output(count:count+n2-1);
count=count+n2;
S=ones(n,m);
S(:)=output(count:count+nm-1);
count=count+nm;
K=zeros(m,n)
K(:)=output(count:count+nm-1)
count=count+nm
ea=zeros(n,n);
for i=1:n
 ea(i,i)=output(count)+j*output(count+1);
 count=count+2;
end
va=zeros(n,n);
for jj=1:n
 for i=1:n
  va(i,jj)=output(count)+j*output(count+1);
  count=count+2;
 end
end
```

Bibliography

1. Robinson, Jeffrey D. A Linear Quadratic Regulator Weight Selection Algorithm for Robust Pole Assignment: MS Thesis, AFIT/GAE/ENG/90D-23. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1990.

2. Huckabone, Captain Thomas C. An Algorithm for Robust Eigenstructure Assignment Using the Linear Quadratic Regulator: MS Thesis, AFIT/GAE/91D-7. School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, December 1991.

3. McRuer, Duane, Irving Ashkenas and Dunstan Graham. Aircraft Dynamics and Automatic Control. Princeton University Press, New Jersey, Princeton, New Jersey, 1973.

4. Reid, J. Gary. Linear Systems Fundamentals: Continuous and Discrete, Classic and Modern. McGraw Hill Publishing Company, New York, New York, 1983.

5. Tischler, M. B. Digital Control of Highly Augmented Combat Rotorcraft. NASA Technical Memorandum 88346, May 1987.

6. Takahashi, M. D. A Flight-Dynamic Helicopter Mathematical Model with a Single Flap-Lag-Torsion Main Rotor. NASA Technical Memorandum 102267, February 1990.

7. Helicopter Stability and Control. Flight Test Manual No. 105, Naval Air Test Center, Patuxent River, Maryland, November 1983, Preliminary Edition.

8. Aeronautical Design Standard, Handling Qualities Requirements for Military Rotorcraft. ADS-33C, United States Army Aviation Systems Command, St. Louis, Missouri, August 1989.

9. Myers, Michael R., Freddie J. Mills, Floyd L. Dominick, Gary L. Skinner and R. Scott Mair. Preliminary Airworthiness Evaluation Prototype MH-60K Helicopter (Handling Qualities). TECOM Report No. 4-CO-230-000-020, United States Army Technical Test Center, Edwards AFB, California, August 1991.

10. Osder, Stephen and Donald Caldwell. "Design and Robustness Issues for Highly Augmented Helicopter Controls," AIAA Guidance, Navigation and Control Conference, Volume 3, Session Paper No. 91-2751: pages 1370-1380 (AUG 1991).

11. Operator's Manual, UH-60A and EH-60A Helicopter, Technical Manual TM-55-1520-237-10, United States Army Aviation Systems Command, St. Louis, Missouri, January 1988.

12. Garrard, William L. and Bradley S. Liebst. "Design of a Multivariable Helicopter Flight Control System for Handling Qualities Enhancement," Journal of the American Helicopter Society, Volume 35: pages 23-30 (OCT 1990).

13. Moore, B. C. "On the Flexibility Offered by State Feedback in Multivariable Systems Beyond Closed Loop Eigenvalue Assignment," IEEE Transactions on Automatic Control, Volume AC-21: pages 689-692 (OCT 1976).

14. Ridgely, D. Brett and Siva S. Banda. Introduction to Robust Multivariable Control. AFWAL-TR-85-3102, Control Dynamics Branch, Flight Dynamics Laboratory, Wright-Patterson AFB, Ohio, February 1986.

15. MATLAB™ User's Manual. The MathWorks, Inc., South Natick, Massachusetts, August 1988.

16. Floyd, Captain R. M. LOGLIB User's Manual: A Description of Computer Routines for Use in Linear Systems Studies. School of Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, May 1984.

17. IMSL User's Manual: Math/Library. IMSL, Inc., Houston, Texas, Volume 1, Chapters 1-2, Version 1.0 (APR 1987).

18. Kuester, James L. and Joe H. Mize. Optimization Techniques with FORTRAN. McGraw Hill Book Company, New York, 1973.

19. Anderson, Brian D. O. and John B. Moore. Optimal Control, Linear Quadratic Methods. Prentice Hall Inc., New Jersey, 1990.

20. Strang, Gilbert. Linear Algebra and Its Applications, Harcourt Brace Jovanovich, San Diego, California, 1988.

21. Kreindler, Eliezer and Anthony Jameson. "Conditions for Nonnegativeness of Partitioned Matrices," IEEE Transactions on Automatic Control, February 1972.

22. Safanov, Michael G. and Michael Athans. "Gain and Phase Margin for Multiloop Regulators," IEEE Transactions on Automatic Control, Volume AC-22: p. 173-179 (APR 1977)

23. Hoh, Roger, _et al_. _Proposed Airworthiness Design Standard: Handling Qualities Requirements for Military Rotorcraft_. Technical Report no. 1194-2, Systems Technology, Inc., Hawthorne, California, December 1985.

24. _VAX FORTRAN: Language Reference Manual_. Digital Equipment Corporation, Mynard, Massachusetts, June 1988.

25. _Introduction to AFIT Educational Computer Services_, Air Force Institute of Technology, May 1991.

Vita

Captain Dempsey D. Solomon was born on 9 July 1960 in Dallas, Texas. He graduated from Mount Pleasant High School in Mount Pleasant, Texas in 1978. He then attended the U.S. Military Academy, at West Point, New York, graduating with a Bachelor of Science degree in 1982. Captain Solomon was commissioned into the U.S. Army in May 1982 as an Aviation Logistics Officer. He completed rotary and fixed wing flight training in January 1984. In 1988, Captain Solomon completed the U.S. Naval Test Pilot School at Patuxent River, Maryland. He served as an Experimental Test Pilot in the U.S. Army Aviation Technical Test Center (USAATTC) at Fort Rucker, Alabama and performed developmental testing on UH-60A/L, CH-47D, OH-58D, AH-64A and non-standard fixed wing aircraft. He has accumulated over 1800 flight-hours in 45 different aircraft during his career. Captain Solomon completed his assignment at Fort Rucker as the Company Commander at USAATTC prior to entering the School of Engineering, Air Force Institute of Technology (AFIT), in May 1991. Following graduation from AFIT, Captain Solomon will be assigned to the 160th Special Operations Aviation Regiment, 101st Airborne Division, at Fort Campbell, Kentucky where he will serve as the Test Coordinator for the U.S. Army's Special Operation's Aircraft program.

Permanent Address: 230 College Street
Pittsburg, Texas 75686

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE December 1992	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE HELICOPTER FLIGHT CONTROL SYSTEM DESIGN USING THE LINEAR QUADRATIC REGULATOR FOR ROBUST EIGENSTRUCTURE ASSIGNMENT	5. FUNDING NUMBERS
6. AUTHOR(S) Dempsey D. Solomon, Captain, USA	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology, WPAFB OH 45433-6583	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GAE/ENY/92D-17
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This thesis applies modern, multi-variable control design techniques, via a FORTRAN computer algorithm, to U.S. Army helicopter models in hovering flight conditions. Eigenstructure assignment and Linear Quadratic Regulator (LQR) theory are used to achieve enhanced closed loop performance and stability characteristics with full state feedback. The addition of cross coupling weights to the standard LQR performance index is specifically addressed. A desired eigenstructure is chosen with a goal of reduced pilot workload via performance characteristics and modal decoupling consistent with current helicopter handling qualities requirements. Cross coupling weighting is shown to provide greater flexibility in achieving a desired closed loop eigenstructure. While the addition of cross coupling weighting is shown to eliminate stability margin guarantees associated with LQR methods, the modified algorithm can achieve a closer match to a desired eigenstructure than previous versions of the program while maintaining acceptable stability characteristics.

14. SUBJECT TERMS Eigenstructure Assignment; Linear Quadratic Regulator; Cross Coupling; Helicopter Control Design	15. NUMBER OF PAGES 96
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL